

# Moving Particles: a parallel optimal Multilevel Splitting method with application in quantiles estimation and meta-model based algorithms

Clément WALTER<sup>a,b,\*</sup>

<sup>a</sup>CEA, DAM, DIF, F-91297 Arpajon, France

<sup>b</sup>Université Paris Diderot, 5 rue Thomas Mann, 75013 Paris, France

---

## Abstract

Considering the issue of estimating *small* probabilities  $p$ , *ie.* measuring a *rare* domain  $F = \{\mathbf{x} \mid g(\mathbf{x}) > q\}$  with respect to the distribution of a random vector  $\mathbf{X}$ , Multilevel Splitting strategies (also called *Subset Simulation*) aim at writing  $F$  as an intersection of *less rare* events (nested subsets) such that their measures are conditionally easily computable. However the definition of an *appropriate* sequence of nested subsets remains an open issue.

We introduce here a new approach to Multilevel Splitting methods in terms of a move of particles in the input space. This allows us to derive two main results: (1) the number of samples required to get a realisation of  $\mathbf{X}$  in  $F$  is drastically reduced, following a Poisson law with parameter  $\log 1/p$  (to be compared with  $1/p$  for naive Monte-Carlo); and (2) we get a parallel optimal Multilevel Splitting algorithm where there is indeed no subset to define any more.

We also apply result (1) in quantile estimation producing a new parallel algorithm and derive a new strategy for the construction of first Design Of Experiments in meta-model based algorithms.

**Keywords:** Rare event simulation, MCMC, Sequential Monte-Carlo, Subset Simulation

---

## 1. Introduction

*Context.* Extreme events simulation and quantification come from the need to insure that undesirable events will not appear. Typically such events are failure of industrial critical systems, *ie.* systems for which failure is regarded as a massive catastrophic situation, found in sectors like nuclear safety, aerospace, etc. In this context one could either want to estimate a probability of failure or to define a threshold to insure security with a given confidence. Usually the system is a "black box" whose output determines safety/failure domains.

Formally, let  $\mathbf{X}$  be a random vector with values in  $\mathbb{R}^d$  and  $g$  be a measurable function from  $\mathbb{R}^d$  to  $\mathbb{R}$  defining the failure domain  $F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > q\}$ , we seek to measure  $F$  with respect to the distribution of  $\mathbf{X}$ :  $P[\mathbf{X} \in F] = \mu^X(F) = p$  or to find back  $q$  given  $p$ .

Two reasons why this calculation is not obvious is the order of magnitude of the probability (say  $p < 10^{-5}$ ) and the computational time of  $g$  output, from couples of hours to several months. In this framework efficiency of the algorithm (precision, global computational time) with respect to good statistical properties of the estimator are of great interest.

Let us first introduce several techniques used so far. Then we will present our new algorithm and its applications in probability and quantile estimation.

*Modified Monte-Carlo algorithms.* A comprehensive review of Monte-Carlo methods can be found in [28]. On the one hand Importance Sampling [26, 2] modifies the distribution of  $\mathbf{X}$  to lower the variance of the naive Monte-Carlo estimator; unfortunately the search for an appropriate change of probability is not obvious. In particular it is known that the optimal change depends on the quantity of interest and is thus not directly available.

On the other hand Multilevel Splitting methods consider the failure domain as a finite intersection of nested subsets for which conditional probabilities are *not too small* and so more easily computable: let  $(F_k)_k$  be a finite sequence of nested subsets ( $F_0 = \mathbb{R}^d$ ) such that  $F = \bigcap_k F_k$ , one can write:

$$P[g(\mathbf{X}) > q] = \mu^X(F) = \prod_k \mu^X(F_k \mid F_{k-1})$$

---

\*Corresponding author. Tel.: +33 1 69 26 66 75; Cel.: +33 6 65 16 53 97  
Email address: clement.walter@cea.fr (Clément WALTER)

20 In these algorithms the two main issues are the conditional sampling and the subsets definition. The idea of splitting an event  $F = \{g(\mathbf{X}) > q\}$  with a sequence of  $(q_m)_m$  such as  $F$  can be written as an intersection of nested subsets appeared in the mid 1950's (from Kahn&Harris [20] and Rosenbluth&Rosenbluth [27]). Then Au&Beck [4] brought it to rare event estimation. An in-depth review of these techniques can be found in Glasserman *et al.* [16]. This algorithm was further improved by Cérou *et al.* [7] linking it with Feynman-Kac formulae and Cérou and 25 Guyader [8] who proposed a method to adaptively select the conditional probabilities. Concerning the conditional simulations, Del Moral *et al.* [9] introduced reversible transition kernels. Finally Cérou *et al.* [6] showed that these algorithms were optimal when all conditional probabilities were equal and that an adaptive choice of levels leads to bias in the estimators. Recently Guyader *et al.* [17] proposed a limit case where the conditional probabilities are fixed to  $1 - 1/N$ , given  $N$  the size of the working population. While they showed that this choice is optimal in terms 30 of computational efficiency, it also disables parallel computation possibility and eventually makes this algorithm longer in practice if multicore computers are available.

*Meta-model based algorithms.* As modified Monte-Carlo methods seen above still require an important number of samples and do not allow for full parallelisation, meta-model based algorithms propose to spend the computational budget in fitting a surrogate model to the expensive-to-evaluate function  $g$  and then to use it instead of the *true* 35 function to compute probability estimation with usual methods [14, 5, 13]. Thus these strategies highly depend on the *quality* of the Design of Experiments (DoE) and especially on their ability to predict the boundary between safety and failure domains, *ie.* to explore the input space close to the boundary. While space-filling strategies recommend to sample uniformly in the input space (see [29] or [13] chapter 2 for a review of these methods) and thus depend a lot on the dimension of the input space, DoE generated according to the distribution of  $\mathbf{X}$  are unlikely 40 to visit the failure domain because of the order of magnitude of the failure probability.

*Effective computing time.* As mentioned previously, the function of interest  $g$  is assumed to be highly demanding in computational time and the number of calls to  $g$ , *ie.* the number of samples used for an estimator, is limited. In this context parallel algorithms (see for examples reference books [21] or [12]) are of great interest. Basically they allow for generating samples for an estimator in a parallel way, *ie.* that one can get as many samples as available 45 "computers" for the time of one. Thus, to increase the precision of an estimator one can simply use more computers, *ie.* multi-core processors, to get more samples without making the estimation longer. We will refer to the number of calls to the limit-state function made by one computer as the effective computing time of an algorithm.

*Main results.* We introduce here a new approach to Multilevel Splitting in terms of a move of particles from an initial random state to the failure domain. This approach brings two main results: first the number of samples 50 needed to get a realisation of  $\mathbf{X}$  in the failure domain follows a Poisson law with parameter  $\log 1/p$ , this is to be compared with a classical Geometric law with parameter  $p$  for naive Monte-Carlo; then we get the full parallelisation of the optimal sequential algorithm described by Guyader *et al.* [17], which turns it into the best Multilevel Splitting algorithm in terms of effective computing time, resolving the issues of choosing a sequence of  $(q_m)_m$  or selecting a cut-off probability for the adaptive construction. This new point of view also allows us to propose a modified 55 version of Guyader *et al.* quantile estimator with a reduced bias and parallel computation.

In the context of meta-model based algorithms (that require DoE with points close to the boundary between safety and failure domains), we use this approach to get a first DoE embedding failing samples while limiting drastically the number of calls to the limit-state function, which depends linearly on the dimension:

$$N_{DoE} = d + 1 + N_{\text{fail}} \log(1/p)$$

with  $N_{DoE}$  the size of the first DoE,  $d$  the dimension of the input space and  $N_{\text{fail}}$  the final number of points in the failure domain.

## 2. Getting into the failure domain

### 2.1. Introduction

60 The idea of trying to go *as fast as possible* into the failure domain comes from the need to get failing samples in several methods, from Importance Sampling [3] to meta-model based algorithm. In this latter case it is well noticed [14] that without a first DoE embedding failing samples, the learning of the failure domain is complicated and the final probability estimator rather poor. On the other hand it was tried to merge Multilevel Splitting methods and meta-model based algorithms to increase the precision of conditional probabilities estimation while making easier 65 the learning of the failure domain [5, 23]. The paradox was that the final DoE was indeed far too dense in *a posteriori* useless regions.

Then we came up with the idea of *stopping* to try computing the probability estimation on-the-go but only keeping the *moving* part of these algorithms. Finally, Guyader *et al.* work [17] brings to us the theoretical framework to derive this algorithm.

70 The problem can be defined as follows: let  $\mathbf{X}$  be a random variable with values in  $\mathbb{R}^d$ ,  $d \in \mathbb{N}^*$ ,  $\mu^X$  its distribution and  $g$  a measurable function from  $\mathbb{R}^d$  to  $\mathbb{R}$  such that the *cdf* of  $g(\mathbf{X})$  is continuous.

We first introduce the algorithm in the ideal case, *ie.* the case where we know how to sample from any distribution when required, and we will then present two implementations to be used depending on the goal (probability and quantile estimation or building of first DoE).

## 75 2.2. Ideal case

In this section we consider that it is possible to sample from any distribution when required; thus it is said *ideal*.

### 2.2.1. Move of one particle

**Algorithm 1** (Move of one particle).

–  $q_0 = -\infty$

80 – *Iteration over  $m$ :*

+ Sample  $\mathbf{X} \sim \mu^X(\cdot \mid g > q_m)$

+ Evaluate  $g$ :  $q_{m+1} = g(\mathbf{X})$

Thus, Algorithm 1 can be seen as a move of a particle from an initial random place along the levels of  $g$  and the sequence of  $(q_m)_m$  has indeed an interesting statistical behaviour. Let  $F_g$  be the *cdf* of  $g(\mathbf{X})$  and  $\Lambda$  the integrated hazard function:  $\Lambda(y) = -\log(1 - F_g(y))$ ; the following results are based on [17]:

**Theorem 1.** *The random variables  $(T_m)_{m \geq 1} = (\Lambda(q_m))_{m \geq 1}$  are distributed as the successive arrival times of a Poisson Process with parameter 1.*

**Corollary 1.** *The number of moves needed to get a realisation of  $\mathbf{X}$  in a domain  $F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > q\}$ ,  $q \in \mathbb{R}$ , with probability measure  $\mu^X(F) = p$  follows a Poisson law with parameter  $\log 1/p$ .*

90 Proofs of these results are given in the appendix.

### 2.2.2. Move of $N$ particles

We now detail how to move a bunch of particles all together and show that it is indeed exactly the same as moving particles all separately. The following algorithm is almost the one proposed by Guyader *et al.* [17] in the context of Multilevel Splitting algorithms, but presented in the scope of moving particles; in particular we do not sort the  $(q_m^j)_j$  at a given iteration  $m$  to keep the particles tractable.

**Algorithm 2** (Move of  $N \in \mathbb{N}^*$  particles).

–  $\mathbf{q}_0 = (q_0^1, \dots, q_0^N) = (-\infty, \dots, -\infty)$

– *Iteration over  $m$ :*

+  $\mathbf{q}_{m+1} = \mathbf{q}_m$

100 +  $i_m = \underset{i}{\operatorname{argmin}}(q_m^i)$

+ Sample  $\mathbf{X}_{i_m} \sim \mu^X(\cdot \mid g > q_m^{i_m})$

+ Evaluate  $g$ :  $q_{m+1}^{i_m} = g(\mathbf{X}_{i_m})$

At time  $m = N$   $(\mathbf{X}_1, \dots, \mathbf{X}_N)$  is an iid sample of the distribution  $\mu^X$ . Afterwards the move of  $N$  particles all together is made sequentially by moving at each step the last one. Once again we study the properties of the sequence of  $(\mathbf{q}_m)_m$ . With the same notations and hypothesis as in section 2.2.1, we have (proofs in the appendix):

**Theorem 2.** *The random variables  $(T_m)_{m \geq N} = (\Lambda(q_m^{i_m}))_{m \geq N}$  are distributed as the successive arrival times of a marked Poisson Process with parameter  $N$*

**Corollary 2.** *The number of moves needed to get  $N$  realisations of  $\mathbf{X}$  in a domain  $F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > q\}$ ,  $q \in \mathbb{R}$ , with probability measure  $\mu^X(F) = p$  follows a Poisson law with parameter  $N \log 1/p$ .*

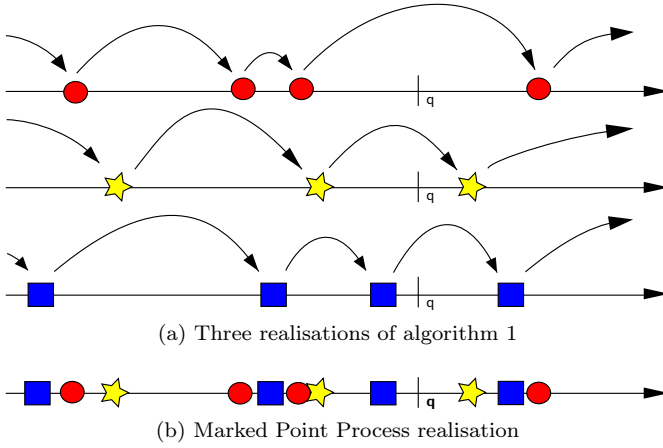


Figure 1: Sketch of algorithms 1 and 2

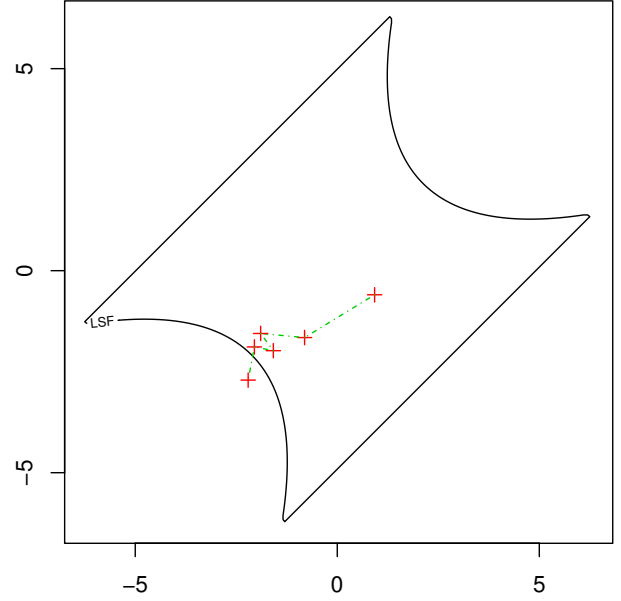


Figure 2: Move of a particle

### 2.3. Practical implementation of algorithms 1 and 2

*Simulating conditional distributions.* We address here the issue of generating according to the conditional law  $\mu^X(\cdot \mid g > q)$  for a given  $q$ . A general idea is to use convergence properties of an ergodic Markov Chain to its unique invariant probability to sample from a given distribution. Assuming  $\mu^X$  has a pdf  $f_X$ , it means we intend to generate a Markov Chain with stationary distribution  $\mathbb{1}_{g>q}f_X$ .

This implementation is especially simple when a reversible transition kernel  $K$  is available. Guyader *et al.* [17] presents two ways of getting it, referred as the Metropolis-Hastings method and a direct construction of the reversible kernel. We will then refer to the use of such a kernel starting from a given  $\mathbf{x}$  as  $K(\mathbf{x}, \cdot)$ .

**Algorithm 3** (Metropolis-Hastings transition Kernel [18]).

- Generate  $\mathbf{W}$  a standard multivariate Gaussian or symmetric Uniform distribution over a compact set of  $\mathbb{R}^d$
- Calculate  $\mathbf{X}^* = \mathbf{x} + \sigma\mathbf{W}$
- Calculate Metropolis-Hastings ratio:  $\rho = \min\left(1, \frac{f_X(\mathbf{X}^*)}{f_X(\mathbf{x})}\right) \mathbb{1}_{g(\mathbf{X}^*)>q}$
- Accept the transition with probability  $\rho$ , return  $\mathbf{x}$  otherwise

**Algorithm 4** (Direct construction for standard Gaussian input space).

- Generate  $\mathbf{W}$  a standard multivariate Gaussian sample
- Calculate  $\mathbf{X}^* = \frac{\mathbf{x} + \sigma\mathbf{W}}{\sqrt{1 + \sigma^2}}$
- Calculate  $g(\mathbf{X}^*)$ ; return  $\mathbf{X}^*$  if  $g(\mathbf{X}^*) > q$ ,  $\mathbf{x}$  otherwise

*The burn-in and starting points parameters.* Because the goal is to reach the stationary state of the Markov Chain from a degenerated distribution, several transitions have to be done to insure independence between the starting point and the final sample and adequacy with the targeted distribution. This number of transitions is referred to as a *burn-in* parameter and denoted by  $T$ . Eventually the last generated sample is kept. In theory one can start from any point provided the *burn-in* is large enough but it is profitable to start with a point already following the targeted distribution as *burn-in* will then serve only independence purpose.

*Choice of the transition Kernel.* The examples presented Section 5 are all defined in the standard Gaussian space. In this context we will use the direct construction of the transition Kernel method (Algorithm 4) with  $\sigma = 0.3$ . The *burn-in* parameter is set to 20 (usual value as in [6] or [17]).

*Technical improvements* . Usual methods select a starting point randomly amongst points already following the targeted distribution. It appears that in practice it is particularly inappropriate to start from a point directly generated by the sample to move, called here a *son*. In this context we adapt the random selection of the starting point for a moving particle to avoid all its sons and their possible replicas (if at one time the  $T$  transitions were refused and so the new particle is simply a copy of the starting point). If at a given iteration there is no such starting point available, it starts from one of those and a move is counted only if at least one transition is accepted.

Furthermore if one has several cores available, it is possible to move simultaneously the  $k$  last particles (sequential parallelisation of algorithm 2). One recalls here that particles move independently and that moving a bunch of  $N$  particles only allows for a better approximation of the stationary law of Markov Chains. Thus as soon as there are *enough* particles *above* the one to be moved, one can make the moves in parallel. Especially,  $\chi^2$  tests were carried out in the examples of Section 5 on the numbers of iterations and they showed that the Poisson feature is very robust even for small  $N$ , *ie.* when the size of the population in which starting points is selected is small.

*Use of a meta-model.* When the targeted distribution depends on the function of interest  $g$ , the Metropolis-Hastings sampling makes indeed the number of calls to  $g$   $T$  times higher than what it *should* be. This is to insure a good convergence of the Markov Chain to the stationary distribution. If one only wants to *move* without any estimation of a statistical quantity, for instance to build a first DoE embedding failing samples for meta-model based algorithms, an approximation may be sufficient. In this scope we suggest the use of a cheap surrogate model to accept and/or reject transitions, true limit-state function being used only a limited number of times to *control* the evolution of a particle. This will be illustrated Section 6.

### 3. Application to extreme probabilities estimation

#### 3.1. Description of the estimator

In this case given  $q$  we define  $F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > q\}$  and we intend to estimate  $p = P[\mathbf{X} \in F]$ . Let us first consider the ideal case: thanks to corollaries 1 and 2 we know that the sought probability is indeed directly related to the Poisson parameter of the number of moves to get realisations of  $\mathbf{X}$  in  $F$ . Given  $K$  independent realisations  $(M_k)_k$  of a random variable following a Poisson law with unknown parameter  $\lambda$ , we thus propose to use the Maximum Likelihood Estimator of the parameter:

$$\hat{\lambda} = \frac{1}{K} \sum_{k=1}^K M_k$$

In our case we are not interested in  $\lambda$  but  $\exp(-\lambda/N)$ , which leads to the following estimator for the probability:

$$\hat{p} = \exp\left(-\frac{\hat{\lambda}}{N}\right) = \exp\left(-\frac{1}{KN}\right)^{\sum_{k=1}^K M_k}$$

Three remarks here:

- $\sum_{k=1}^K M_k$  follows a Poisson law with parameter  $-KN \log p$  and thus  $K$  and  $N$  are playing symmetric roles, which means that in the ideal case moving particles one by one or in cluster does not change anything.
- Writing  $x = KN$  we notice that making the asymptotic expansion of order 1 of  $\exp(-1/x)$  gives the estimator proposed by Guyader *et al.* [17].
- Furthermore considering  $1 - 1/x$  instead of  $\exp(-1/x)$  enforces an unbiased estimator and gives a lower mean squared error  $E[(\hat{p} - p)^2] = \text{var}[\hat{p}] + (E[\hat{p}] - p)^2$ .

Thus we chose the unbiased estimator for the probability  $p$ :

$$\hat{p} = \left(1 - \frac{1}{KN}\right)^{\sum_{k=1}^K M_k} \quad (1)$$

with  $K$  the total number of algorithms run in parallel,  $N$  the number of particles per algorithm, and  $M_k$  the number of mutations for the  $k$ -th algorithm.

### 3.2. Practical implementation

In practice we can generate random variables  $(M_k)_k$  of the number of mutations to get  $N$  particles in  $F$  and make the Poisson distribution approximation more accurate by increasing  $N$  and  $T$ . Indeed there is a trade-off between the three parameters  $K$ ,  $N$  and  $T$ : the greater  $K$  the faster the parallel estimation but the greater  $N$  and  $T$  the more robust.

Ideally,  $N$  should be as small as possible to allow for a greater parallelisation but regarding the examples (cf Section 5)  $N$  should not be smaller than 10.

Furthermore, given  $N_{\text{call}}$  the total number of calls to the limit-state function and  $T$  the *burn-in* parameter for conditional sampling, we have:

$$N_{\text{call}} = T \{\text{total number of iterations}\} \approx -TKN \log p = \alpha T \frac{-KN}{\alpha} \log p \quad (2)$$

which means that for an equivalent computational budget, one can increase  $T$  as much as one decreases  $N$ . Finally, given  $n_c$  the number of cores available for calculation, the algorithm for estimating a probability writes as follows:

**Algorithm 5** (Algorithm for estimating a probability).

- Launch  $n_c$  times in parallel algorithm 2 with stopping criteria  $\min(\mathbf{q}_m) \geq q$  and  $N \geq 10$
- Count the total number of mutations  $M = \sum_{k=1}^{n_c} M_k$
- Calculate  $\hat{p} = \left(1 - \frac{1}{Nn_c}\right)^M$

### 3.3. Statistical analysis of the estimator

From now on, let us consider that  $N$  particles were moved, regardless of the number of algorithms practically used as this has no impact on ideal theoretical results. We then have the same results as [17] and reader is referred to this article for the proofs of the following results.

*Moments of  $\hat{p}$ .* One can calculate the moments of  $\hat{p}$  using the probability-generating function of a Poisson random variable. We get:

$$\mathbb{E}[\hat{p}] = p; \quad \text{var}[\hat{p}] = p^2 \left( p^{-\frac{1}{N}} - 1 \right) \quad (3)$$

This means this estimator almost achieves the Cramer-Rao bound:  $-p^2 \log p/N$ .

*Confidence interval.* The discrete random variable  $\hat{p}$  follows a Poisson distribution with parameter  $-N \log p$ . Furthermore we get  $\log \hat{p} \stackrel{\mathcal{L}}{\sim} \mathcal{P}(-N \log p) \log(1 - 1/N)$ . Considering  $N \geq 10$  and  $p \leq 10^{-3}$  brings the following approximation:

$$\log \hat{p} \sim \mathcal{N}(\log p, -\log p/N) \quad (4)$$

**Proposition 1.** Given  $\alpha \in [0, 1]$  and  $Z_{1-\alpha/2}$  the quantile of order  $1 - \alpha/2$  of the standard normal distribution:  $\mathbb{P}[-Z_{1-\alpha/2} < \mathcal{N}(0, 1) < Z_{1-\alpha/2}] = 1 - \alpha$ ; an asymptotic confidence interval for  $\hat{p}$  at  $(1 - \alpha)\%$  is given by:

$$\hat{p} \exp \left( -\frac{Z_{1-\alpha/2}^2}{2N} - \sqrt{\Delta} \right) \leq p \leq \hat{p} \exp \left( -\frac{Z_{1-\alpha/2}^2}{2N} + \sqrt{\Delta} \right) \quad (5)$$

with:

$$\Delta = \frac{Z_{1-\alpha/2}^2}{N} \left( \hat{t} + \frac{Z_{1-\alpha/2}^2}{4N} \right) \quad \text{and} \quad \hat{t} = -\log \hat{p}$$

### 3.4. Comparison with Monte-Carlo methods

In this section we compare our algorithm with the ones presented in the introduction. We focus on the number of particles needed to gain a given precision on the estimator and on the computing time required to achieve it.

*Coefficient of variation* . The main difference between [4] and [6] stands in the sampling technique: while [4] uses Markov Chains, [6] uses particles methods developped by [9]. This leads [4] to introduce a coefficient  $\gamma$  which characterises the correlation in between samples from the same Markov Chain (refer to the article for more details).

Considering  $\delta$  the coefficient of variation (CV) of the estimator, we find in the corresponding articles in the first order in  $N$ :

$$\begin{aligned} & - \text{ in [4]: } \delta_{\text{Au}}^2 = \frac{\log p}{\log p_0} \frac{(1+\gamma)(1-p_0)}{Np_0} \\ & - \text{ in [6]: } \delta_{\text{Cer}}^2 = \frac{1}{N} \left( \lfloor \frac{\log p}{\log p_0} \rfloor \frac{1-p_0}{p_0} + \frac{1-r_0}{r_0} \right) \\ & - \text{ in [17]: } \delta_{\text{Guy}}^2 = \frac{-\log p}{N} \end{aligned}$$

For the purpose of our study we will consider the optimal setting  $r_0 \approx p_0$  (like [4]) and drop the  $\gamma$  coefficient to keep the following relation for Multilevel Splitting methods:

$$\delta_{\text{Au}}^2 \approx \delta_{\text{Cer}}^2 \approx \delta_{\text{Guy}}^2 \approx \delta_{\text{MS}}^2 \approx \frac{\log p}{\log p_0} \frac{1-p_0}{Np_0} \quad (6)$$

where  $p_0 = 1 - 1/N$  in  $\delta_{\text{Guy}}$ .

*Computational time.* Given a targeted precision  $\delta$  on the estimator  $\hat{p}$ , we now focus on the computational time required to achieve it. Let  $n_c$  be the number of available cores; one can consider that all operations are cost-free apart from calls to the limit-state function and so computational time reduces to the number of calls to the limit-state function done by each core.

**Naive Monte-Carlo algorithm** In this algorithm there is no dependency between samples and so each core will have to do  $N_{\text{call}}/n_c$  calculations:

$$t_{\text{MC}} = \lceil \frac{N_{\text{call}}}{n_c} \rceil = \lceil \frac{1}{n_c \delta^2 p} \rceil \quad (7)$$

**Multilevel Splitting algorithms** In Multilevel Splitting algorithms, there are  $N$  samples generated initially and then  $N(1-p_0)$  regenerated at each iteration. Thus, the computational time writes as follows:

$$\begin{aligned} t_{\text{MS}} &= \frac{N}{n_c} + \lfloor \frac{\log p}{\log p_0} \rfloor T \left[ \frac{N(1-p_0)}{n_c} \wedge 1 \right] \\ t_{\text{MS}} &\approx \frac{\log p}{\log p_0} \frac{1-p_0}{n_c \delta^2 p_0} + \frac{\log p}{\log p_0} T \left[ \frac{\log p}{\log p_0} \frac{(1-p_0)^2}{n_c \delta^2 p_0} \wedge 1 \right] \end{aligned} \quad (8)$$

Depending on the parametrisation of the algorithm (choice of  $N$  and  $p_0$ , number of cores  $n_c$ ) we will have either  $N(1-p_0) \geq n_c$  and so:

$$t_{\text{MS}} \approx \frac{T(\log p)^2}{n_c \delta^2} \frac{(1-p_0)^2}{p_0 (\log p_0)^2} \quad (9)$$

or  $N(1-p_0) \leq n_c$  and so:

$$t_{\text{MS}} = \frac{\log p}{\log p_0} T \quad (10)$$

Formula (9) is strictly decreasing in  $p_0$  while (10) is strictly increasing. This can eventually suggest an optimal value  $p_0^*$  for  $p_0$  depending on the sought probability  $p$ , the number of cores  $n_c$  and the precision  $\delta$ , it is the solution of:

$$\frac{\log p}{\log p_0^*} \frac{(1-p_0^*)^2}{n_c \delta^2 p_0^*} = \frac{N(1-p_0^*)}{n_c} = 1 \quad (11)$$

Finally we can write:

$$t_{\text{MS}}(p_0) \geq t_{\text{MS}}(p_0^*) = \frac{T(\log p)^2}{n_c \delta^2} \frac{(1-p_0^*)^2}{p_0^* (\log p_0^*)^2} > \frac{T(\log p)^2}{n_c \delta^2} \quad (12)$$

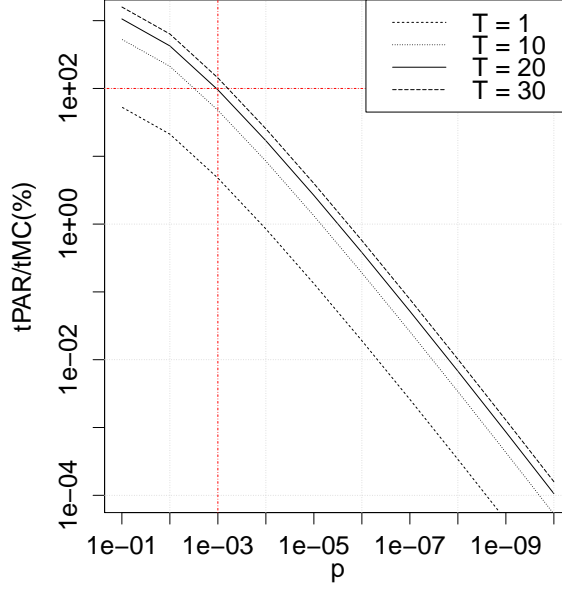


Figure 3:  $t_{\text{par}}/t_{\text{MC}}(\%)$  against  $p$  for different *burn-in* parameters

**Moving Particles algorithm** There we have  $N/n_c$  particles per algorithm 2 and so the "arrival times" of the  $n_c$  algorithms will be distributed as the realisations of  $n_c$  Poisson laws with parameter  $-N/n_c \log p$ .

**Proposition 2.** Let  $t_{\text{par}}$  be the random variable of the effective computing time for full parallel algorithm 5, we have:

$$E[t_{\text{par}}] = \frac{T(\log p)^2}{n_c \delta^2} \left( 1 + \sqrt{\frac{n_c \delta^2}{(\log p)^2}} \sqrt{2 \log n_c} + \frac{1}{T \log 1/p} \right) \quad (13)$$

The additional term due to the full parallelisation would drop in case of sequential parallelisation. Indeed considering the "arrival times" are normally distributed, there will be as many *shorter* than *longer* algorithms comparing to the reference value  $-N/n_c \log p$ . If one can afford sequential parallelisation (cf. Section 2.3), shorter and longer algorithm will compensate each other. So for comparison with classical Sequential Monte-Carlo one can retain:

$$t_{\text{par}} = \frac{T(\log p)^2}{n_c \delta^2} \quad (14)$$

*Conclusion on the estimator.* On the one hand we have found an optimal value of  $p_0$  for classical Multilevel Splitting algorithm in terms of computational time (considering that the only important operation is a call to the limit-state function). On the other hand we see that our approach gives always a better result than the Multilevel Splitting method with the optimal  $p_0$ . In other words, our approach allows for taking  $p_0 \rightarrow 1$  while keeping the parallel computation. Thus this is the optimal way of computing Multilevel Splitting methods.

Furthermore, with standard values of  $\log 1/p \approx 10^1$ ,  $n_c \approx 10^2$  and  $\delta^2 = 10^{-2}$ , we get  $(\log p)^2/(n_c \delta^2) \approx 10^2$ , which means  $t_{\text{par}}$  is multiplied by  $\approx 1.1$  if no sequential parallelisation is applied. As sequential parallelisation can indeed increase computational time consequently, this result is of great interest because it shows that without losing too much time the implementation of Multilevel Splitting methods can be a lot easier.

Finally we compare  $t_{\text{MC}}$  and  $t_{\text{par}}$ :

$$\log \left( \frac{t_{\text{par}}}{t_{\text{MC}}} \right) \underset{p \rightarrow 0}{\sim} \log (Tp(\log p)^2) \quad (15)$$

Figure 3 plots  $t_{\text{par}}/t_{\text{MC}}(\%)$  for several values of  $T$  in logarithmic scale. It is clearly visible that the relation becomes linear when  $p$  becomes small and especially for a standard value of  $T = 20$ , our algorithm makes better than a naive Monte-Carlo as soon as  $p \lesssim 10^{-3}$ .



## 215 4. Application to extreme quantile estimation

### 4.1. Description of the estimator

In this case we define  $F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) > q\}$  and  $p = P[\mathbf{X} \in F]$  and we intend to estimate  $q$ . Let us first suppose we can have the infinite sequence of the events  $(T_m)_m$  of the Poisson Process (with  $T_0 = 0$ ) and derive some results. We focus on the counting random variable  $M_t = \sup\{m \geq 0 \mid T_m \leq t\}$  at time  $t = -\log p$ . We know  
 220 that  $M_t \stackrel{\mathcal{L}}{\sim} \mathcal{P}(-N \log p)$ .

**Lemma 1** (Laws of random variables  $T_{M_t}$  and  $T_{M_t+1}$ ). *Let us denote by  $F_{M_t}$  and  $F_{M_t+1}$  the cdf associated to  $T_{M_t}$  and  $T_{M_t+1}$  respectively. The following result holds:*

$$\forall(\alpha, \beta) \in \mathbb{R}_+^2, P[(T_{M_t+1} - t > \alpha/N) \cap (t - T_{M_t} \geq \beta/N)] = e^{-\alpha} e^{-\beta} \mathbb{1}_{[0; Nt)}(\beta) \quad (16)$$

**Corollary 3.** *The center of the interval  $[T_{M_t}; T_{M_t+1}]$  converges toward a random variable centred in  $t$  with symmetric pdf, ie. :*

$$N \left( \frac{T_{M_t+1} + T_{M_t}}{2} - t \right) \xrightarrow[Nt \rightarrow \infty]{\mathcal{L}} Z \quad (17)$$

with  $Z$  a random variable with pdf  $f_Z(z) = e^{-2|z|}$

From Lemma 1 and Corollary 3 we can define the following quantity of interest:

$$\tilde{q} = \frac{1}{2} q_{M_t} + \frac{1}{2} q_{M_t+1}$$

with  $(q_m)_m = (\Lambda^{-1}(T_m))_m$ . We assume that the pdf  $f$  of  $g(\mathbf{X})$  is continuous at  $q$ ; knowing the law of  $(T_{M_t}, T_{M_t+1})$  and making an asymptotic expansion around  $t$  we can calculate the moments of  $\tilde{q}$  and get:

$$E[\tilde{q}] = q + o\left(\frac{1}{N}\right) \quad (18)$$

$$\text{var}[\tilde{q}] = \frac{p^2}{f(q)^2} \frac{1}{N^2} + o\left(\frac{1}{N^2}\right) \quad (19)$$

### 4.2. Practical implementation

Unfortunately we do not observe the  $(T_m)_m$  but only  $q_m = \Lambda^{-1}(T_m)$  for which we do not have any expression and we cannot observe  $M_t$  nor  $T_{M_t}$  and  $T_{M_t+1}$ . However we know that  $M_t \stackrel{\mathcal{L}}{\sim} \mathcal{P}(-N \log p)$ ; writing  $M = \lceil -N \log p \rceil$ , we then choose as an estimator for  $q$ :

$$\hat{q} = \frac{1}{2} q_{M-1} + \frac{1}{2} q_M \quad (20)$$

Indeed we want to estimate  $q$  by looking at some particular events defined by their rank. Especially if several algorithms are run in parallel, they will be events of the full Process:  $n_c$  computers available brings  $M = \lceil N n_c \log p \rceil$ .  
 225 However to rebuild the full process until event number  $M$  requires to make sure that all realisations  $(T_m)_m$  have overpassed a given time  $t$  and that the total number of events before that time  $t$  will be greater than  $M$ . We propose here two possible ways to achieve this with parallel algorithms: a 2-passes algorithm and a sequential one. The sequential algorithm is optimal considering the number of calls to the limit-state function but allows only for a sequential parallelisation, which can indeed turn it into a longer one's. Denoting by  $(q_m^i)_m$  the sequence of the  
 230 successive minima of algorithm number  $i \in \llbracket 1; n_c \rrbracket$ , the algorithms are:

**Algorithm 6** (A 2-passes algorithm for quantile estimation).

- Do  $n_c$  times algorithm 2 with stopping criteria "number of moves =  $m_0$ " and  $N \geq 10$
- Get the maximal value reached:  $q_{max} = \max_{i=1..n_c} q_{m_0}^i$
- Restart the  $n_c$  algorithms with stopping criteria " $\min(\mathbf{q}_m) \geq q_{max}$ "
- Merge the  $n_c$  sequences  $(q_m^i)_m$  and sort the resulting one; call it  $(q_m)_m$
- Calculate  $\hat{q} = \frac{1}{2} (q_{M-1} + q_M)$

The choice of  $m_0$  will be discussed further section 4.4. One can already notice that depending on this value it will be certain ( $m_0 = \lceil -N \log p \rceil$ ) or not to eventually get enough events.

**Algorithm 7** (A sequential algorithm for quantile estimation).

- 240     $-$  Do  $n_c$  times algorithm 2 with stopping criteria "number of moves = 1" and  $N \geq 10$
- $-$  Get the minimal value reached:  $q_{min} = \min_{i=1..n_c} q_1^i$
- $-$   $k = 1$ . While  $\#\{q_j^i \mid q_j^i \leq q_{min}; i = 1..n_c, j = 1..k\} < M$ 
  - $+ k := k + 1$
  - $+ \text{Restart the } n_c \text{ algorithms with stopping criteria "number of moves = 1"}$
- 245     $+ \text{Get the minimal value reached: } q_{min} = \min_{i=1..n_c} q_k^i$
- $- \text{Merge the } n_c \text{ sequences } (q_m^i)_m \text{ and sort the resulting one; call it } (q_m)_m$
- $- \text{Calculate } \hat{q} = \frac{1}{2} (q_{M-1} + q_M)$

250    The sequential version of the algorithm insures a minimum number of iterations as it stops as soon as there are enough events. However it needs communication between the  $n_c$  algorithms which can finally slow down the whole estimation.

#### 4.3. Statistical analysis of the estimator

*Some general results.* Let  $N$  be the total number of particles and  $m = \lceil -N \log p \rceil$ . We first present some asymptotic results for an estimator  $q_{m+k} = \Lambda^{-1}(T_{m+k})$  with a given  $k$  as  $N \rightarrow +\infty$ . Then we study the property of a linear combination of  $q_{m+k}$  around  $m$ . From now on, let us suppose that  $g(\mathbf{X})$  has cdf  $F$  and pdf  $f$  continuous at  $q$ .

**Proposition 3** (Central Limit Theorem). *If  $f(q) \neq 0$ , then:*

$$\sqrt{N} (q_{m+k} - q) \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f(q)^2}\right) \quad (21)$$

**Proposition 4** (Bounds on bias). *With the same hypothesis as for a naive Monte-Carlo estimator (see for example [1] p.128) and  $f'(q) < 0$  we get the following boundaries for the bias:*

$$\begin{aligned} \mathbb{E}[q_{m+k}] - q &\geq \frac{p}{Nf(q)} \left( \frac{-\log p}{2} \left( 1 + \frac{f'(q)p}{f(q)^2} \right) + k \right) + o\left(\frac{1}{N}\right) \\ \mathbb{E}[q_{m+k}] - q &\leq \frac{p}{Nf(q)} \left( \frac{-\log p}{2} \left( 1 + \frac{f'(q)p}{f(q)^2} \right) + k + 1 \right) + o\left(\frac{1}{N}\right) \end{aligned} \quad (22)$$

**Proposition 5** (Confidence interval). *Writing  $Z_{1-\alpha/2}$  the  $(1-\alpha)\%$  quantile of a standard Gaussian distribution,  $m_- = \lfloor m - Z_{1-\alpha/2} \sqrt{m} \rfloor$  and  $m_+ = \lceil m + Z_{1-\alpha/2} \sqrt{m} \rceil$ , we have:*

$$\mathbb{P}[q \in [q_{m_-}, q_{m_+}]] \xrightarrow[m \rightarrow \infty]{} 1 - \alpha \quad (23)$$

**Proposition 6** (Multidimensional Central Limit Theorem). *Let us now consider the vector  $(q_m, q_{m+1}, \dots, q_{m+k})$ . We can write the following multidimensional central limit theorem:*

$$\sqrt{N} \left[ \begin{pmatrix} q_m \\ \vdots \\ q_{m+k} \end{pmatrix} - q \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right] \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}\left( \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \frac{-p^2 \log p}{f(q)^2} \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix} \right) \quad (24)$$

**Proposition 7.** *The estimator  $\hat{q} = \frac{1}{2}(q_{m-1} + q_m)$  has the following properties:*

$$\sqrt{N}(\hat{q} - q) \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f(q)^2}\right) \quad (25)$$

and:

$$\begin{aligned} \mathbb{E}[\hat{q}] - q &\geq \frac{p}{Nf(q)} \left( \frac{-\log p}{2} \left( 1 + \frac{f'(q)p}{f(q)^2} \right) - \frac{1}{2} \right) + o\left(\frac{1}{N}\right) \\ \mathbb{E}[\hat{q}] - q &\leq \frac{p}{Nf(q)} \left( \frac{-\log p}{2} \left( 1 + \frac{f'(q)p}{f(q)^2} \right) + \frac{1}{2} \right) + o\left(\frac{1}{N}\right) \end{aligned} \quad (26)$$

**Remark 1.** *For exponential tails, ie. distributions with pdf  $f(x) \sim \exp(-(\frac{|x|}{x_0})^k)$  for some  $k \in \mathbb{N}$  and  $x_0 \in \mathbb{R}_+^*$  one have:*

$$\frac{-\log p}{2} \left( 1 + \frac{f'(q)p}{f(q)^2} \right) \underset{p \rightarrow 0}{\sim} \frac{-1}{2} \frac{k-1}{k}$$

with equality if  $k = 1$ . This means that for exponential tails, the bounds on bias simplify as follow:

$$-\frac{1}{2} \frac{p}{f(q)} \left( 2 - \frac{1}{k} \right) \leq \liminf_N \mathbb{E}[\hat{q}] - q \leq \limsup_N \mathbb{E}[\hat{q}] - q \leq \frac{1}{2k} \frac{p}{f(q)} \quad (27)$$

#### 4.4. Comparison with Monte-Carlo methods

As for the probability estimator, we now want to benchmark the computing time of our algorithm given a precision. Unlike in Section 3.4 we can only compare with naive Monte-Carlo as Multilevel Splitting methods are not able to produce quantile estimators and Guyader *et al.* algorithm does not allow for parallelisation.

*Naive Monte-Carlo algorithm.* Naive Monte-Carlo estimator has the following properties:

$$\sqrt{N}(\hat{q}_{MC} - q) \xrightarrow[N \rightarrow \infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{p(1-p)}{f(q)^2}\right) \quad (28)$$

260 with  $N$  the number of samples and  $f$  the pdf of  $g(\mathbf{X})$ .

This brings for the coefficient of variation:

$$\delta^2 = \frac{p(1-p)}{Nf(q)^2} \frac{1}{q^2} + o\left(\frac{1}{N}\right) \approx \frac{p}{Nq^2f(q)^2} \quad (29)$$

and then the computing time:

$$t_{MC} = \left\lceil \frac{N}{n_c} \right\rceil = \left\lceil \frac{p}{q^2f(q)^2\delta^2n_c} \right\rceil \quad (30)$$

*Moving Particles algorithm.* Both versions of the algorithm (6 and 7) produce finally the same estimator and so have the same statistical properties:

$$\delta^2 = \frac{-p^2 \log p}{n_c N q^2 f(q)^2} + o\left(\frac{1}{n_c N}\right) \quad (31)$$

with  $N$  the number of particles per algorithm and  $n_c$  the number of cores.

The computing time of each algorithm depends on the number of iterations done by each parallel algorithm; especially for algorithm 6 this number will be driven by the parameter  $m_0$ .

265 **Number of iterations of Algorithm 6** After the first pass we have indeed  $n_c$  iid random variables with distribution  $\Gamma_{m_0}/N$  random variables for which we consider the maximum  $T_{\max}$ . In the second pass, we move particles until time  $T_{\max}$  and so the total number of iterations follows a Poisson law with parameter  $NT_{\max}$ .

**Proposition 8.** *Let  $N_{iter}$  be the random variable counting the number of iterations of algorithm 6, we have:*

$$\mathbb{E}[N_{iter}] = m_0 + \sqrt{2m_0 \log n_c} \quad (32)$$

We now detail a criteria for choosing  $m_0$ . Indeed, we consider that one can accept to take a risk  $\alpha$  that the Poisson Process does not go further enough. Let  $M_t$  be the counting variable of the marked Poisson Process at a time  $t$  and  $m = \lceil -n_c N \log p \rceil$  the targeted number of events, the criteria writes as follows:

$$\mathbb{P}[M_{T_{\max}} \leq m] \leq \alpha \quad (33)$$

**Proposition 9** (Choice of  $m_0$ ). *With the previous notations and  $t = -\log p$ , we have:*

$$m_0 = \lceil Nt + \beta^2/2 - \beta\sqrt{\Delta}/2 \rceil \quad (34)$$

with:

$$\beta = b_{n_c} - \frac{\log \log 1/\alpha}{\sqrt{2 \log n_c}}$$

$b_{n_c}$  being the localisation parameter of the Gaussian law in the framework of Extreme Value Theory:  $b_{n_c} = \sqrt{2 \log n_c} - (\log \log n_c + \log 4\pi) / 2\sqrt{2 \log n_c}$  and

$$\Delta = \beta^2 + 4Nt$$

**Remark 2.** *The targeted value  $\alpha$  should not be set too small as the approximation of Gamma laws with Gaussian distributions is not correct for rare events. However we know that taking  $m_0 = \lceil -N \log p \rceil$  ensures a sufficient number of iterations because of  $n_c \lceil -N \log p \rceil \geq \lceil -n_c N \log p \rceil$ . Furthermore, if after Algorithm 6 the number of events is not sufficient, one can restart it for the number of missing events.*

**Proposition 10** (Expectation of  $N_{\text{iter}}$  in the 2-passes algorithm). *With this value of  $m_0$ , we have:*

$$\mathbb{E}[N_{\text{iter}}] \approx -N \log p + \sqrt{2N \log 1/p \log n_c} \quad (35)$$

**Number of iterations of Algorithm 7** In this sequential algorithm there is no parameter  $m_0$  to set. Writing  $T_{\min}(k)$  the minimum of  $k$  iid random variables with distribution  $\Gamma_k/N$  and  $M_t$  the counting random variable of the marked Poisson Process at a given time  $t$ , the algorithm can be written as follows:

$$\text{While } M_{T_{\min}(k)} \leq \lceil -n_c N \log p \rceil, k = k + 1$$

**Proposition 11** (Expectation of  $N_{\text{iter}}$  in the sequential algorithm).

$$\mathbb{E}[N_{\text{iter}}] = -N \log p + \sqrt{-2N \log p \log n_c} \quad (36)$$

**Computing time of Moving Particles algorithm** Since the expectations of the number of iterations are approximately the same for both algorithms (Algorithm 6 and Algorithm 7), we can finally derive the computing time of the quantile estimator as a function of the coefficient of variation:

$$t_{\text{par}} \approx \frac{T}{n_c} \left( \frac{p \log p}{\delta q f(q)} \right)^2 \left( 1 + \frac{1}{T \log 1/p} + \delta \gamma(q) \sqrt{2n_c \log n_c} \right) \quad (37)$$

with  $\gamma(q) = \frac{q f(q)}{-p \log p}$ .

Eventually we can compare this time to the one from a naive Monte-Carlo estimator:

$$\log \left( \frac{t_{\text{par}}}{t_{\text{MC}}} \right) = \log (Tp(\log p)^2) + \log \left( 1 + \frac{1}{T \log 1/p} + \delta \gamma(q) \sqrt{2n_c \log n_c} \right) \quad (38)$$

**Remark 3** (Order of magnitude of  $\gamma(q)$ ). *While there is no general result on the order of magnitude of  $\gamma(q)$ , it can be shown that in a lot of cases it remains small. For instance if one considers von Mises distributions, ie. distributions such that the cdf  $F$  has the following representation:*

$$1 - F(q) = \bar{F}(q) = c \exp \left( - \int_z^q \frac{1}{a(t)} dt \right) \quad (39)$$

with  $c$  a given positive constant and  $a(\cdot)$  the auxiliary function of  $F$ :  $a = \bar{F}/f$  (see [15] Definition 3.3.18), one obtains:

$$\frac{1}{\gamma(q)} = \frac{-p \log p}{q f(q)} = -\frac{a(q)}{q} \log c + \frac{a(q)}{q} \int_z^q \frac{dt}{a(t)}$$

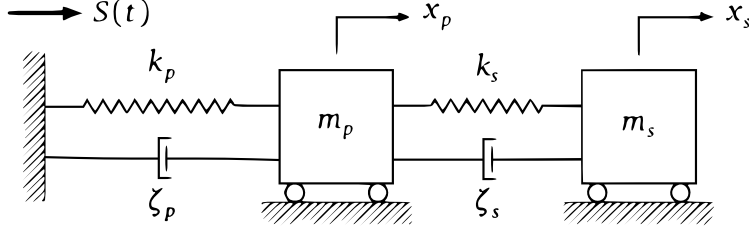


Figure 4: A 2 degrees of freedom damped oscillator (from [13])

In this equality, the first term tends to 0 (see [15] Proposition 3.3.24) and the second one can be bounded from below by  $1 - z/q$  for any  $z \in \mathbb{R} \mid \forall x \in \mathbb{R}, x \geq z \Rightarrow a'(z) \geq 0$ , which means  $\gamma(q) \in [0, 1]$ .

Furthermore for exponential tails, ie.  $f(x) \underset{+\infty}{\sim} \exp(-(\frac{|x|}{x_0})^k)$  for some  $k \in \mathbb{N}$  and  $x_0 \in \mathbb{R}_+^*$  one can write (see [15] Example 3.3.23):

$$p = -\frac{f(q)^2}{f'(q)}(1 + o(1)) = \frac{e^{-q^k}}{kq^{k-1}}(1 + o(1))$$

which brings:

$$1/\gamma(q) = \frac{-p \log p}{q f(q)} \propto \frac{1}{k} \left( 1 - \frac{\log(kq^{k-1})}{q^k} + o\left(\frac{1}{q^k}\right) \right) \underset{p \rightarrow 0}{\sim} \frac{1}{k}$$

Eventually one can conclude that  $\gamma(q)$  remains low.

275 As for the probability estimator there is an extra term driven by  $\sqrt{n_c \log n_c}$  and eventually the order of magnitude remains the same and the quotient  $t_{\text{par}}/t_{\text{MC}}$  is always driven by  $p$  for small values of  $p$ .

## 5. Examples

### 5.1. Presentation of the examples

#### 5.1.1. Watermarking detection

This example is the one used by C  rou *et al.* [6] and Guyader *et al.* [17] to show the properties of their algorithms. Let  $d \in \mathbb{N}^*$  be the dimension of the input space and  $\mathbf{u}$  be a unit vector in  $\mathbb{R}^d$ ; the failure domain is regarded as the interior of a double cone of axis  $\mathbf{u}$  (see [24]):

$$F = \{\mathbf{x} \in \mathbb{R}^d \mid \Phi(\mathbf{x}) = \frac{|\mathbf{x}^T \mathbf{u}|}{\|\mathbf{x}\|} > q\} \quad (40)$$

The analytic relation between  $p$  and  $q$  writes as follows:

$$p = \mathbb{P}(\Phi(\mathbf{X}) > q) = 1 - F(q) = 1 - G\left(\frac{(d-1)q^2}{1-q^2}\right)$$

280 with  $F$  the *cdf* of  $\Phi(\mathbf{X})$  and  $G$  the *cdf* of a Fisher variable with  $(1, d-1)$  degrees of freedom (see [17]).

#### 5.1.2. A two-degrees-of-freedom damped oscillator

This example sketched in Figure 4 was first proposed by Der Kiureghian and De Stefano [11] and then used by Bourinet *et al.* [5] and Dubourg [13].

It is a two degrees of freedom damped oscillator characterised by masses  $m_p$  and  $m_s$ , spring stiffnesses  $k_p$  and  $k_s$ , natural frequencies  $\omega_p^2 = k_p/m_p$  and  $\omega_s^2 = k_s/m_s$  and damping ratios  $\zeta_p$  and  $\zeta_s$ . Igusa and Der Kiureghian [19] showed that the mean-squared relative displacement of the secondary spring under a white noise base acceleration with intensity  $S_0$  writes as follows:

$$\mathbb{E}[x_s^2] = \pi \frac{S_0}{4\zeta_s \omega_s^2} \frac{\zeta_a \zeta_s}{\zeta_p \zeta_s (4\zeta_a^2 + \theta^2) + \gamma \zeta_a^2} \frac{(\zeta_p \omega_p^3 + \zeta_s \omega_s^3) \omega_p}{4\zeta_a \omega_a^4}$$

with  $\gamma = m_s/m_p$ ,  $\omega_a = (\omega_p + \omega_s)/2$ ,  $\zeta_a = (\zeta_p + \zeta_s)/2$  and  $\theta = (\omega_p - \omega_s)/\omega_a$

Finally, Der Kiureghian and De Stefano [11] showed that the limit-state function could write under reasonable approximation as follows:

$$g(\mathbf{x}) = F_s - p k_s \sqrt{\mathbb{E}[x_s^2]} \quad (41)$$

with  $F_s$  the force capacity of the secondary spring and  $p$  a peak factor here set to 3 as in [13].

Table 1 presents the probabilistic model used. As it uses lognormal distributions and reversible kernel is defined

Variable	Mean	CV (%)
$m_p$	1.5	10
$m_s$	0.01	10
$k_p$	1	20
$k_s$	0.01	20
$\zeta_p$	0.05	40
$\zeta_s$	0.02	50
$F_s$	{15; 21.5; 27.5}	10
$S_0$	100	10

Table 1: Stochastic model of the oscillator

in the standard space a conversion is done before each call to the limit-state function.

## 5.2. Estimation of failure probability

*Watermarking detection.* Here we set  $d = 20$ ,  $q = 0.95$  and we try to estimate  $p = 1 - G\left(\frac{(d-1)q^2}{1-q^2}\right) = 4.704 \cdot 10^{-11}$ .

*A two-degrees-of-freedom damped oscillator.* Failure is defined as  $g(\mathbf{x}) < 0$ . There is no analytical expression available to benchmark our algorithm and so reference values have been calculated with reference estimators as described Table 2.

### 5.2.1. Effect of the choice of $N$ and $n_c$ for a total of 1000 particles

The purpose of this part is to precise the minimal acceptable value for  $N$  as the smaller  $N$  the faster the whole algorithm. The following configurations have been tested, expressed as " $n_c \times N$ ": "1x1000", "10x100", "20x50", "50x20", "100x10", "200x5", "500x2" and "1000x1". Results are shown in Figure 5 as boxplots of 100 simulations, whiskers extending to the extreme values. The reference value is displayed with the red dashed line, and in the case of the 2 d-o-f oscillator 95% confidence interval is displayed as well with the black dashed lines (see Table 2). For all examples it appears that for  $N = 10$  and above the estimations are almost the same.

### 5.2.2. Effective computing time

In this section we look at the effective computing time of the estimators, *ie.* the maximum number of calls to the limit-state function made by each of the  $n_c$  algorithms for a given configuration. Especially we intend to check the consistency of formula (13) while number of calls not only depends on the *burn-in* parameter  $T$  and the number of mutations, but also the number of rejected mutations, *ie.* transitions where the starting point was a replica of the moving particle and all transitions were refused (see 2.3); this situation especially arises when the number of particles gets small ( $N \leq 5$ ).

As for the probability, results are displayed as boxplots overs 100 simulations in Figure 6. The red dots show the theoretical value given by equation (13). One can see that apart from the two last configurations of watermarking detection examples, these values are in good agreement with the empirical results.

$F_s$	Method	Nbr of points	Proba	CV
15	Monte-Carlo brut	2000000	$4.8015 \cdot 10^{-3}$	0.01018
21.5	Monte-Carlo brut	10000000	$4.34 \cdot 10^{-5}$	0.048
27.5	Subset Simulation	4000000	$3.745 \cdot 10^{-7}$	0.0286

Table 2: Reference values for the two-degrees-of-freedom damped oscillator

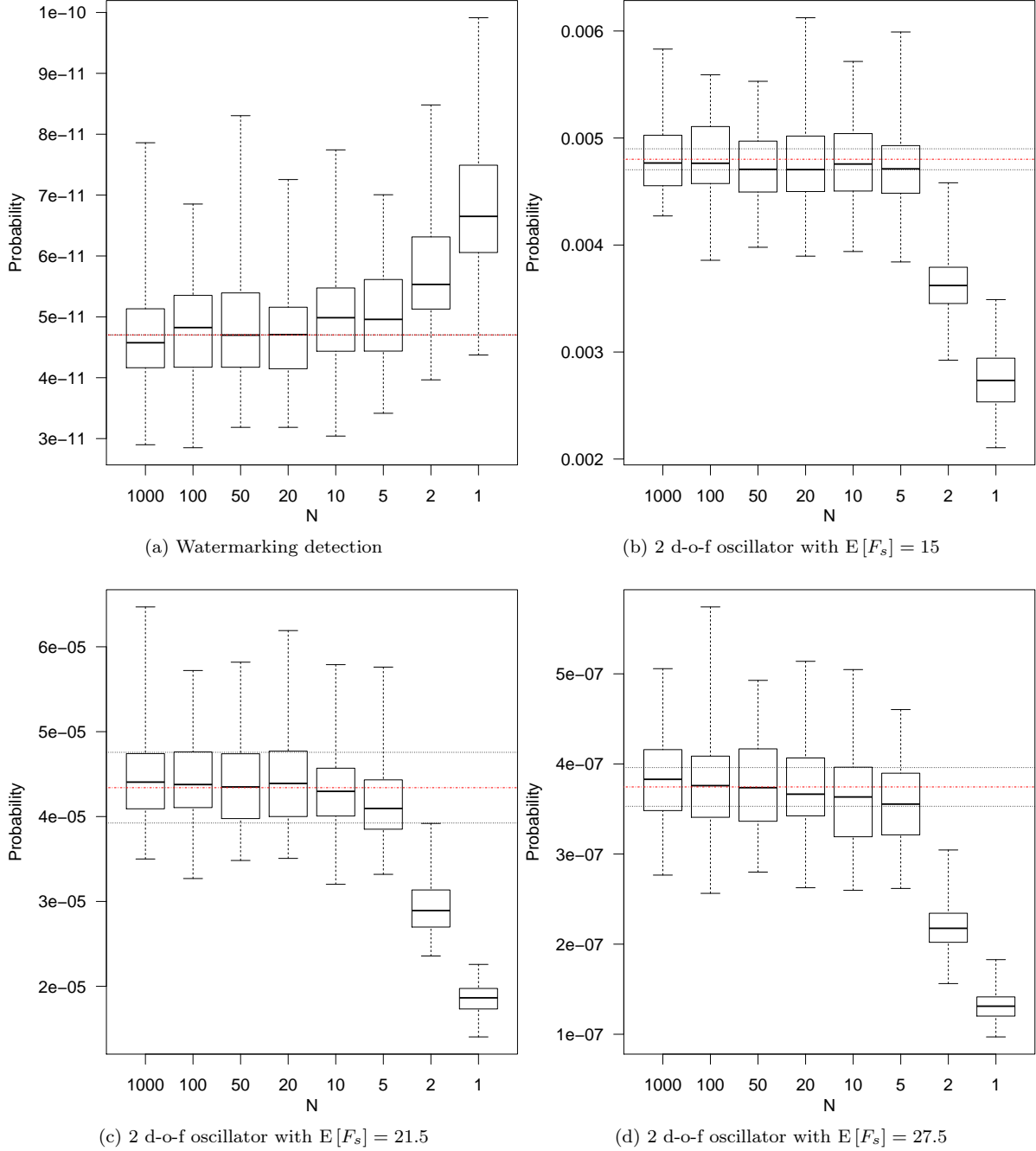


Figure 5: Boxplots of the probability estimator over 100 simulations

### 310 5.2.3. Conclusion on the estimator

The moving particle point of view allows us to define a new estimator for extreme probabilities. This estimator can be related to Multilevel Splitting methods and especially had already been proposed in this framework by Guyader *et al.* [17]. Nevertheless we can now compute it in a parallel way which makes it more efficient than any Multilevel Splitting strategy; especially gain is approximately of 50% in terms of computing time for an equivalent precision comparing to *Subset Simulation* as described by [4]. Comparing to naive Monte-Carlo, a standard choice of  $T = 20$  for the *burn-in* parameter makes our algorithm better as soon as  $p \lesssim 10^{-3}$  and gain increases as  $1/p$ .

315 Practically speaking, the minimal number of particles to be considered in each algorithm 2 seems to depend on the limit-state function and taking  $N \geq 10$  appears to be a conservative choice.

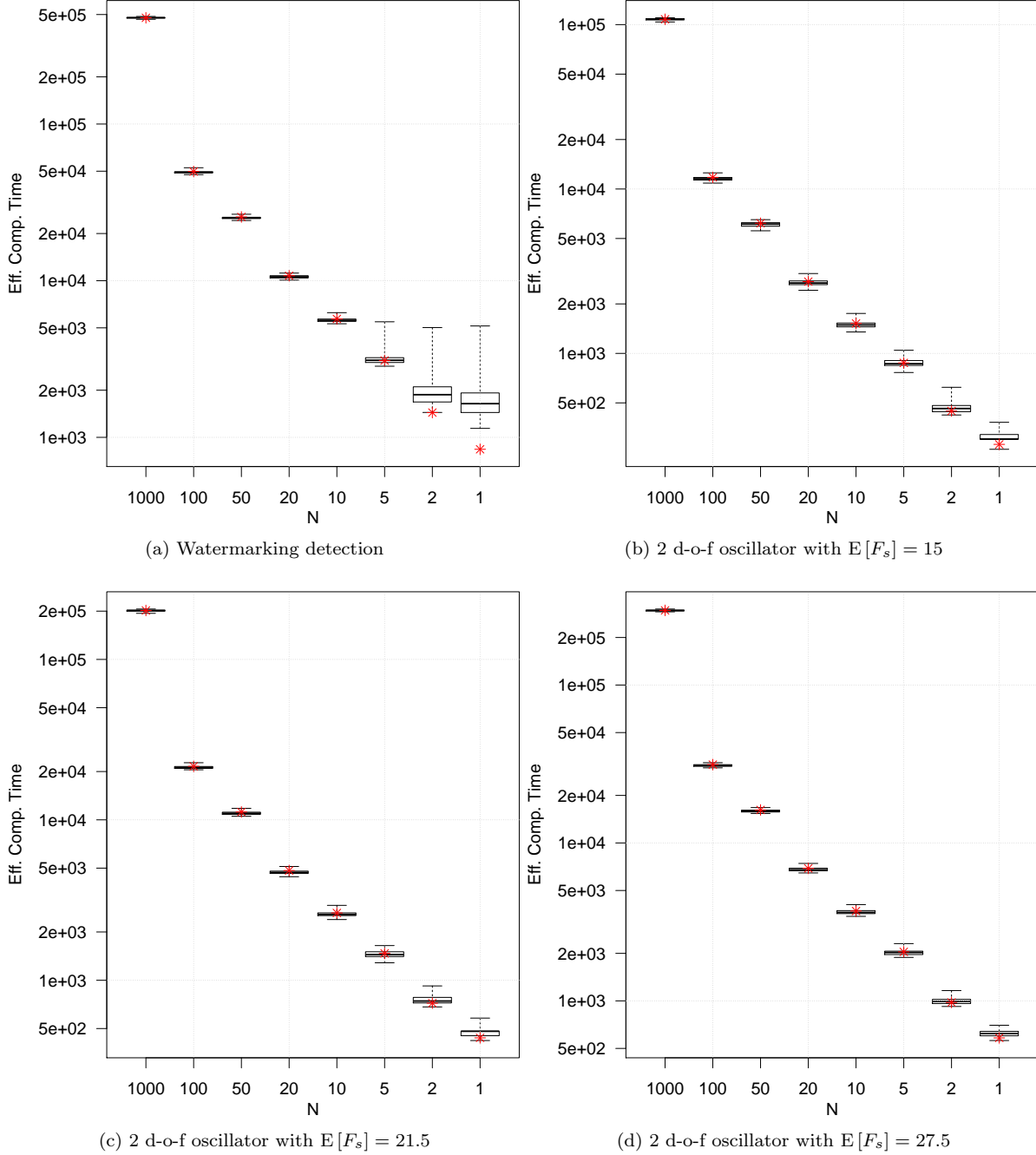


Figure 6: Effective computing time of the probability estimator over 100 simulations

### 5.3. Estimation of quantiles

320 We use the example presented section 5.1.1 to qualify our algorithm. Unlike probability estimator, which was indeed the same as Guyader *et al.* one's (we only intended to check parallelisation capacity) we have presented here above a new estimator. We thus set  $p = 4.704 \cdot 10^{-11}$  and try to find back  $q = 0.95$ .

#### 5.3.1. Effect of the choice of $N$ and $n_c$ for a total of 1000 particles

325 The context is the one presented section 5.2.1. Results are displayed in Figure 7 as boxplots over 100 simulations, whiskers extending to the extreme values and reference value is added to the plot with a dashed line. Once again for  $N \geq 10$  the estimations seem to be almost the same. One could expect to get the same type of results as for the probability because the algorithm are intrinsically the same, *ie.* a move of particles. Thus a higher estimation



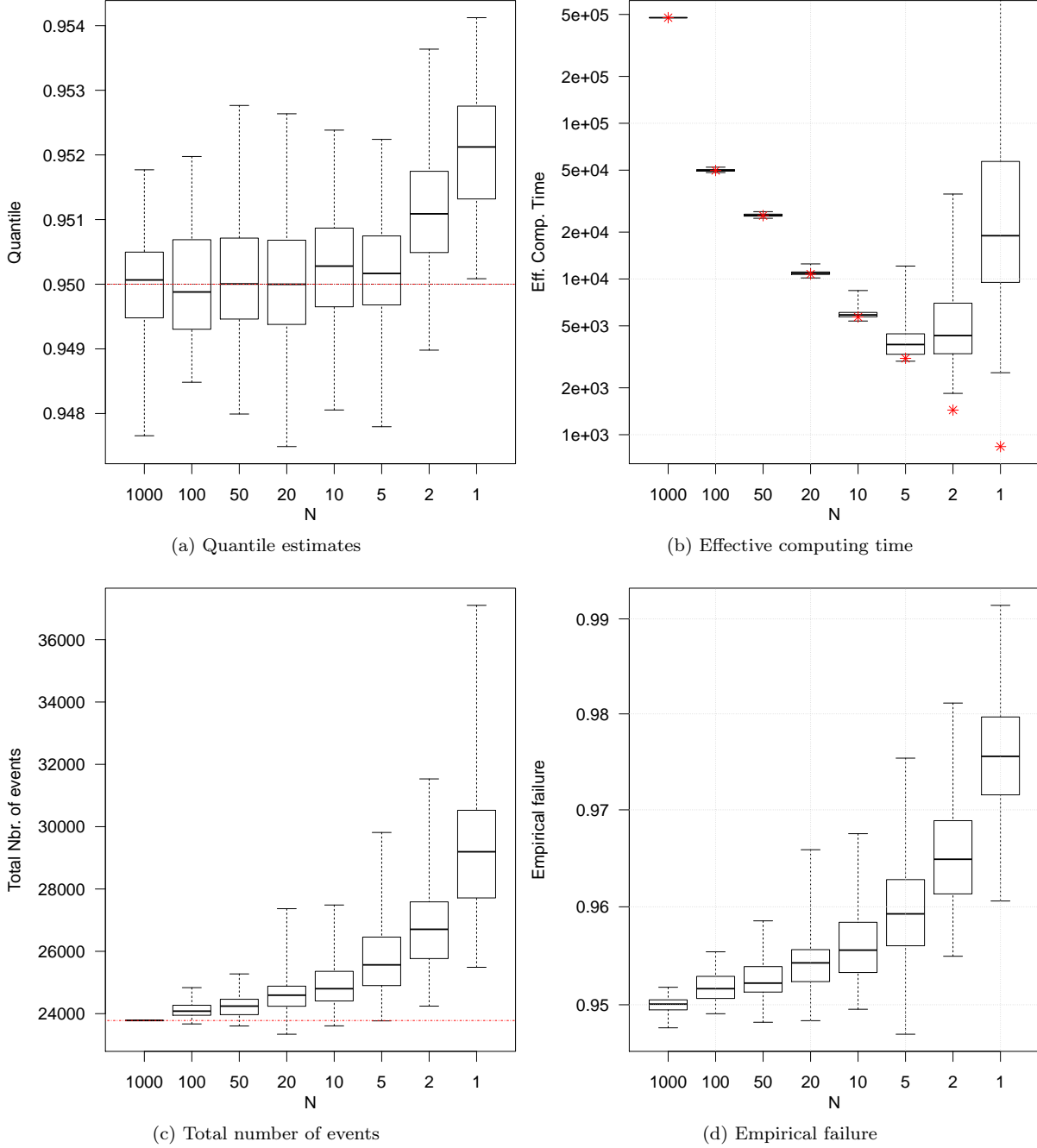


Figure 7: Statistics on quantile estimator over 100 simulations

of the probability means a too low number of iterations, *ie.* that particles are moving *too fast* which will directly produce an overestimation of the quantile.

### 5.3.2. Effective computing time

As for the probability estimator we intend to validate formula (37). The results are presented in Figure 7b and show a good agreement with the formula apart from the *extreme* cases ( $N \leq 5$ ). This is because the smaller the population, the higher the intermediate failure level (cf. Figure 7d) and so greater the number of transitions to stop the algorithm. As for small populations ( $N \leq 5$ ) the generation step does not work well, more transitions means more generation steps with the rejection of all transitions, *ie.* "useless" calls to the limit-state function.

We also check the total number of mutations as this should be ideally equal to the targeted one's:  $m =$

$[-n_c N \log p]$  and in practice as close as possible. Especially we have accepted here to take a risk  $\alpha = 5\%$  (see equation (9)) not to have enough sample at the end of the algorithm. In Figure 7c we can see that in some cases the algorithm did not produce enough events. Especially the number of "too short" algorithms are 5, 4, 3, 3 and 1 respectively. On a total of 100 simulations this is in good agreement with the parameter  $\alpha$  set to 5%.

### 5.3.3. Conclusion on the estimator

Unlike the probability estimator, which was indeed the one proposed by Guyader *et al.*, the moving particles point of view allows us to define a new estimator with a reduced bias. Statistical properties are only asymptotic as for naive Monte-Carlo estimator but confidence intervals can be computed without any estimation of the *pdf*. In terms of computing time, it has the same asymptotic properties as the estimator of probabilities. As far as we know, it is the first parallel alternative to naive Monte-Carlo and thus allows for a massive gain for quantile estimation.

## 6. Application to the construction of first Design of Experiments

In this section we intend to apply the moving particles point of view to the building of DoE for meta-model based algorithms. Especially it means we only keep the *moving part* while dropping the constraint on *exact* sampling.

### 6.1. Meta-modelisation and first DoE

*The fixed-time framework.* While Algorithm 5 seen above is optimal compared to Multilevel Splitting methods, it still requires a lot of calls to the limit-state function. Furthermore this number of calls directly depends on the unknown parameter  $p$  while in a practical context it is more often about using a given number of calls in the best way.

In this scope meta-model based algorithms use this computational budget to fit a surrogate model and then use it instead of the true function to estimate probabilities or quantiles.

*The importance of the first DoE.* A major issue with these techniques is the control of the *fidelity* of the surrogate model to the true function; especially one have to make sure the input space is explored enough to get a *good* approximation of the failure domain. The sampling strategies are usually twofold: first an initial sampling which aims at giving an overall knowledge of the function, then an iterative refinement of the surrogate model according to a given criterion. The *quality* of the first DoE is of great importance as it is highly unlikely that the refinement step be able to find back the boundary between safety and failure domain if no failing points are known. Furthermore the computing time constraint does not allow for a revision of the strategy if unsuccessful.

*What is it to learn about.* It is noteworthy that regardless of the kind of meta-model techniques used (classification through Support-Vector Machine like Bourinet *et al.* algorithm <sup>2</sup>SMART [5] or regression as in AKMCS [14] or MetaIS [13]), the meta-model is always *finally* used as a classification, directly or through the probability of being in the failure domain (MetaIS). Thus there is no real need to get a good knowledge of the limit-state function over the whole input space but only "around" the boundary delimiting  $F$ : it does not matter if regression is accurate in safety and failure domains, the prediction only needs to belong to the right domain.

Nevertheless usual strategies of *Space filling* do not take benefit from this analysis and look at a global learning of the limit-state-function in the input space. Unfortunately this brings to direct exponential dependency on the dimension of the input space. On the other hand density-based DoEs (like in AKMCS) use the distribution of the input variable to sample from and select some points by clustering. While it does not depend on the dimension of the input space anymore, it is very unlikely to produce failing point as by definition  $p$  is supposed to be small. An ideal strategy would indeed produce only pairs of points across the boundary; there is no need to get a sharp estimation of  $g$  on both domains.

These observations lead us to propose a new strategy for first DoE based on algorithm 1 as it allows to get failing points *quickly*.

### 6.2. Adaptation of algorithm 1 to the construction of first DoE

*First DoE.* As pointed out Section 2.3 we can afford an approximation of the conditional sampling if we only intend to *move*. The first things we need is to define a meta-model with a minimal-sized DoE, then we will be able to generate as many Markov Chains as desired number of failing points. Usual techniques use Maximum Likelihood Estimation to estimate the hyper-parameters of a meta-model, which requires at least  $d + 1$  points, given  $d$  the dimension of the input space (for dimensional ranges and standard deviation). We thus expect a final number of calls to the limit-state-function:

$$N_{\text{final}} \approx (d + 1) + N_{\text{fail}}(-\log p) \quad (42)$$

380 with  $N_{\text{fail}}$  the number of failing points desired and so a linear dependency on the dimension. Finally we choose here to use Gaussian Process Regression (Kriging) and without any expert knowledge we fix the trend to be equal to the failure threshold: estimating it would implicitly exclude some unknown parts of the input space as this estimation done with the data will be lower than  $q$ .

**Algorithm 8** (Algorithm for getting a first DoE).

- 385 1. Sample  $d + 1$  points according to  $\mu^X$  and calculate  $g$
2. Learn a first meta-model  $\tilde{g}$  with trend =  $q$
3. Do  $N_{\text{fail}}$  times
- Sample  $\mathbf{X}_1$  with respect to  $\mu^X$
  - Evaluate  $g$ :  $g(\mathbf{X}_1) = y_1$ ;  $m = 1$
  - 390 - Train the meta-model
  - While  $y_m < q$ 
    - $\mathbf{X}_{m+1} = \mathbf{X}_m$ ;  $y_{m+1} = y_m$
    - Do  $T$  times
      - +  $\mathbf{X}^* \sim K(\mathbf{X}_{m+1}, \cdot)$
      - 395 +  $\tilde{g}(\mathbf{X}^*) = y^*$
      - + If  $y^* > y_{m+1}$ ,  $y_{m+1} = y^*$  and  $\mathbf{X}_{m+1} = \mathbf{X}^*$
    - Evaluate  $g$ :  $g(\mathbf{X}_{m+1}) = y_{m+1}$
    - Train the meta-model
    - If  $y_{m+1} < y_m$ ,  $\mathbf{X}_{m+1} = \mathbf{X}_m$  and  $y_{m+1} = y_m$
    - 400 -  $m = m + 1$

One could notice that the *burn-in* parameter  $T$  is not exactly a proper *burn-in* any more because the current threshold changes during the loop. As the main goal here is to move to the failure domain, there is a trade-off between huge moves and keeping close to the already explored input space. In fact this *pseudo burn-in* allows for a move to "the end of the information".

### 405 6.3. Examples

We test our algorithm on the previous limit-state functions and on some others presented below. We focus on the number of calls to the limit-state-function which should be distributed according to equation (42). For all examples, 10 failing points were asked to the algorithm and  $T$  is set to 20.

*A two-dimensional four branches serial system.* This example has been originally proposed by P.-H. Waarts [30] and is defined as follows in the standard space:

$$g : \mathbf{x} \in \mathbb{R}^2 \mapsto \min \left( 3 + \frac{1}{10}(x_1 - x_2)^2 - \frac{1}{\sqrt{2}} |x_1 + x_2|, \frac{7}{\sqrt{2}} - |x_1 - x_2| \right) \quad (43)$$

*A parabolic limit-state function.* A. Der Kiureghian and T. Dakessian [10] proposed this parabolic limit-state function whose equation is in the standard space:

$$g : \mathbf{x} \in \mathbb{R}^2 \mapsto b - x_2 - \kappa(x_1 - \varepsilon)^2 \quad (44)$$

with  $b = 5$ ,  $\kappa = 0,5$  and  $\varepsilon = 0.1$

A *concave failure domain*. This function from R. Rackwitz [25] defines a concave failure domain in  $\mathbb{R}^d$ ,  $d$  being the dimension of the input space. It is defined in a lognormal input space ( $d$  independent random variables with mean  $\mu = 1$  and standard deviation  $\sigma$ ) and thus normal-lognormal transformation is done before each call to the limit-state function. The original function in the lognormal input space writes as follows:

$$g : \mathbf{x} \in \mathbb{R}^d \mapsto d + a\sigma\sqrt{d} - \sum_{i=1}^d x_i \quad (45)$$

with  $a = 3$  and  $\sigma = 0.2$ , which becomes here:

$$g : \mathbf{u} \in \mathbb{R}^d \mapsto d + a\sigma\sqrt{d} - \sum_{i=1}^d \exp\left(-\frac{1}{2}\log(1 + \sigma^2) + u_i\sqrt{\log(1 + \sigma^2)}\right) \quad (46)$$

Since the function is defined for any  $d \in \mathbb{N}^*$  we try it with  $d = 2$ ,  $d = 20$  and  $d = 50$  to study the behaviour of the algorithm depending on the dimension of the input space.

Figure 8 shows three steps of Algorithm 8: the first DoE, the first Markov Chain and the final DoE with the meta-model. We notice that the second move took a wrong direction and proposed  $\mathbf{X}_3$  was *farther* from the failure domain. In this case, it started back from  $\mathbf{X}_2$  while keeping this information for the train of the meta-model.

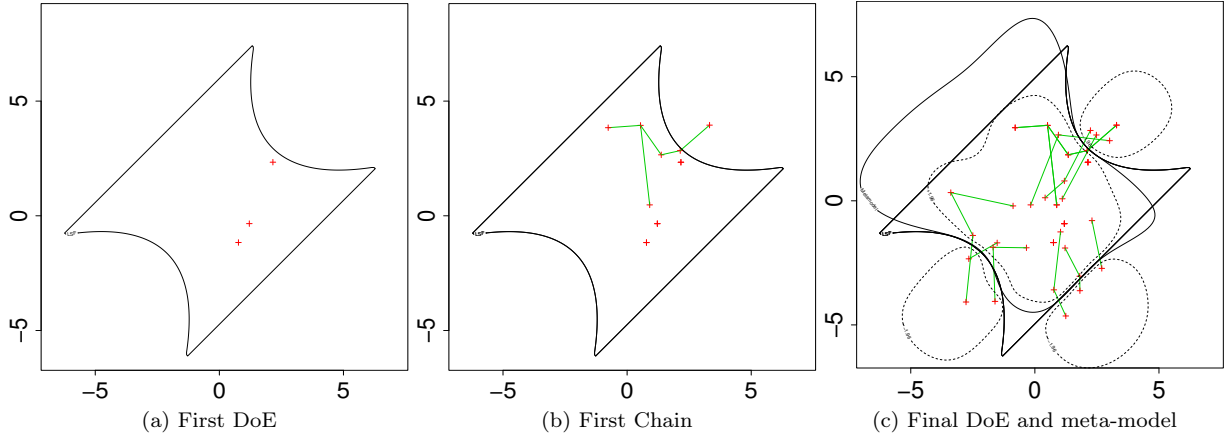


Figure 8: Three steps of algorithm 8 on a serial function

**Results.** In this part we intend to compare the effective number of calls to the one given by the equation (42). To compute the theoretical value, we use an estimate of the failure probability obtained with one run of algorithm 5 with  $N = 5000$  particles; it is indeed not necessary to get highly precise probability estimates as we focus on consistency between theory and practice only. We also present the number of iterations per particle to see how the progressive learning of the function modifies the behaviour of the algorithm.

Both results are summed up in Table 3 (adequacy to the equation (42)) and Table 4 (evolution of the number of iterations per particle).

Finally practical results show a good behaviour of the algorithm on different situations and it appears to be quite versatile. Apart from specific Watermarking test-case the number of iterations does not seem to depend on the rank of the failing point. In this latter case the algorithm struggled to get the two first failing points and then went a lot faster into the failure domain: it is to suppose that it found a specific path and then put all the points in the same *area*.

#### 6.4. Perspectives

The adaptation of Algorithm 1 to the construction of first DoE seems to be a good mean to get into the failure domain. Once *there*, two questions remain open:

- how to prevent the algorithm from moving the particles all at the same place? in other words: how to make the meta-model *understand* that it is not necessary to go back to regions it has already visited? Self-avoiding walks could be used here.

Fonction	dim	Probability	Theor. $N_{\text{call}}$	Pract. $N_{\text{call}}$	Theor. $N_{\text{iter}}$	Pract. $N_{\text{iter}}$
Waarts	2	$2.275 \cdot 10^{-3}$	64.25	34	5.83	3.1
Kiureghian	2	$2.946 \cdot 10^{-3}$	61.27	31	6.09	3
Concave	2	$4.821 \cdot 10^{-3}$	56.35	36	5.35	3.3
Concave	20	$2.273 \cdot 10^{-3}$	81.87	53	6.09	3.2
Concave	50	$1.861 \cdot 10^{-3}$	113.87	82	6.29	3.1
Oscillator 15	8	$4.802 \cdot 10^{-3}$	62.75	48	5.38	3.9
Oscillator 21.5	8	$4.46 \cdot 10^{-5}$	109.75	78	10.02	6.9
Oscillator 27.5	8	$3.76 \cdot 10^{-7}$	156.94	113	14.79	10.4
Watermarking	20	$4.704 \cdot 10^{-11}$	258.80	259	23.78	23.8

Table 3: Getting into the failure domain with algorithm 8

Fonction	Theor. $N_{\text{iter}}$	Pract. $N_{\text{iter}}$
Waarts	5.83	{3 ; 4 ; 4 ; 3 ; 3 ; 2 ; 3 ; 3 ; 3 ; 3 }
Kiureghian	6.09	{4 ; 3 ; 2 ; 3 ; 4 ; 4 ; 3 ; 2 ; 3 ; 2 }
Concave	5.35	{2 ; 3 ; 5 ; 4 ; 3 ; 4 ; 3 ; 4 ; 3 ; 2 }
Concave	6.09	{3 ; 2 ; 4 ; 3 ; 4 ; 3 ; 4 ; 3 ; 4 ; 2 }
Concave	6.29	{4 ; 2 ; 4 ; 3 ; 3 ; 3 ; 3 ; 5 ; 2 ; 2 }
Oscillator 15	5.38	{7 ; 2 ; 3 ; 2 ; 5 ; 4 ; 4 ; 5 ; 2 ; 5 }
Oscillator 21.5	10.02	{7 ; 6 ; 5 ; 9 ; 5 ; 11 ; 6 ; 7 ; 7 ; 6 }
Oscillator 27.5	14.79	{15 ; 9 ; 8 ; 7 ; 10 ; 15 ; 9 ; 15 ; 6 ; 10 }
Watermarking	23.78	{34 ; 61 ; 16 ; 25 ; 16 ; 14 ; 13 ; 18 ; 20 ; 21 }

Table 4: Number of iterations per particle in algorithm 8

- once in the failure domain, what should be an *efficient* strategy to refine the meta-model? Indeed usual refinement strategies appear here like a backtracking as there are all global strategies and would not use the local accuracy of the meta-model to the limit-state function close to the failing point. Some strategies trying to follow the boundary from the failing points seem to show promising results.

## Appendix

**Proof of Theorem 1** First  $\Lambda$  is an increasing function from  $\mathbb{R}$  to  $\mathbb{R}_+$  and the sequence  $(q_m)_m$  is increasing and so the sequence  $(T_m)_m$  is.

It remains to show that it is a Poisson Process with parameter 1, which means by definition that inter-arrival times are independent and follow an exponential law with parameter 1. Considering  $m \in \mathbb{N}$  (with the convention  $q_0 = -\infty$  and  $T_0 = 0$ ) we have:

$$\begin{aligned}
T_{m+1} - T_m &= \Lambda(q_{m+1}) - \Lambda(q_m) = -\log(\mathbb{P}[g(\mathbf{X}) > q_{m+1}]) + \log(\mathbb{P}[g(\mathbf{X}) > q_m]) \\
&= -\log\left(\frac{\mathbb{P}[g(\mathbf{X}) > q_{m+1}]}{\mathbb{P}[g(\mathbf{X}) > q_m]}\right) \\
T_{m+1} - T_m &= -\log(\mathbb{P}[g(\mathbf{X}) > q_{m+1} \mid g(\mathbf{X}) > q_m])
\end{aligned}$$

Let  $\mathcal{F}_m$  be the  $\sigma$ -algebra generated by  $(T_j)_{j \leq m}$  and  $F_m$  be the *cdf* of  $g(\mathbf{X})$  when the distribution of  $\mathbf{X}$  is  $\mu^{\mathbf{X}}(\cdot \mid g > q_m)$ . Knowing  $\mathcal{F}_m$ ,  $F_m$  is the *cdf* of  $q_{m+1}$  and thus  $F_m(q_{m+1})$  follows a uniform law on  $[0; 1]$ . Finally we get:

$$\begin{aligned}
\forall t \in \mathbb{R}_+, \mathbb{P}[T_{m+1} - T_m < t \mid \mathcal{F}_m] &= \mathbb{P}[-\log(1 - F_m(q_{m+1})) < t \mid \mathcal{F}_m] \\
&= \mathbb{P}[F_m(q_{m+1}) < 1 - \exp(-t) \mid \mathcal{F}_m] \\
\forall t \in \mathbb{R}_+, \mathbb{P}[T_{m+1} - T_m < t \mid \mathcal{F}_m] &= 1 - \exp(-t)
\end{aligned} \tag{.1}$$

Thus the inter-arrival times are independent and follow an exponential law with parameter 1 and  $(T_m)_m$  is a Poisson Process.

**Proof of Corollary 1** In this context we focus on the time  $t = -\log(1 - F(q)) = -\log p$ . The number of events before  $t$  equals the number of moves to get the first realisation of  $\mathbf{X}$  in  $F$  and thus follows a Poisson law with parameter  $t = \log 1/p$ .

**Proof of Theorem 2** There is only one move per iteration and so each particle is indeed moved according to Algorithm 1, *ie.* conditionally to its current position and the times only depend on the position. Given  $i_m$  the index of the particle realising the  $m^{\text{th}}$  minimum:  $\forall m \geq N, i_m = \underset{i}{\operatorname{argmin}} q_m^i$  (if several particles achieve the minimum  $i_m$  is chosen uniformly amongst the indices), we build:

$$T_m \mapsto (T_m, i_m)$$

445 which is a marked Point Process in  $\mathbb{R}_+^* \times \mathbb{N}^*$  (see [22] for more details about marked Point Processes). The processes on each mark are independent Poisson Process with parameter 1 and algorithm 2 is finally a marked Poisson Process with parameter  $\sum 1 = N$

**Proof of Corollary 2** The Poisson process is marked and so the particles move indeed independently. As expressed by Corollary 1 each particle has to move  $M \stackrel{\mathcal{L}}{\sim} \mathcal{P}(\log 1/p)$  times. Finally the total number of moves is the sum of  
450 the number of moves of each particle and thus follows a Poisson law with parameter  $N \log 1/p$ .

**Proof of Proposition 2** Let  $\lambda$  be the parameter of the Poisson laws:  $\lambda = -N/n_c \log p$ . In the extreme value theory framework, we are interested in the so called location parameter  $b_n$  which drives the mean of the maximum of  $n$  iid random variables with *cdf*  $F$ . It is the solution of the equation:

$$b_n = F^{-1} \left( 1 - \frac{1}{n} \right)$$

For the Normal distribution, we have:

$$b_n = \sqrt{2 \log n} + \frac{\log \log n + \log 4\pi}{2\sqrt{2 \log n}} \sim \sqrt{2 \log n}$$

Furthermore, we know that Normal approximation of a Poisson distribution with parameter  $\lambda$  is valid in the range  $(-\sqrt{\lambda}, \sqrt{\lambda})$ . Here, this means that as soon as  $\sqrt{2 \log n_c} < \sqrt{\lambda}$ , we can use the approximation of the Poisson law by a Normal one's to calculate the constant. The practical values of  $N/n_c \approx 10^1$ ,  $-\log p \approx 10^1$  and  $n \approx 10^2$  allows us to make the approximation:

$$\mathcal{P}(\lambda) \sim \mathcal{N}(\lambda, \lambda)$$

Let  $N_{\max}$  be the random variable of the maximum of  $n_c$  iid standard Gaussian variables. The total number of calls is the sum of the  $N/n_c$  initial calls to the limit-state function and the number of iterations. Thus we have:

$$\mathbb{E}[t_{\text{par}}] = \mathbb{E} \left[ T(N_{\max} \sqrt{\lambda} + \lambda) + N/n_c \right] \approx T \left( \frac{N}{n_c} \log 1/p + \sqrt{\frac{N}{n_c} \log 1/p \sqrt{2 \log n_c}} \right) + \frac{N}{n_c}$$

and so in terms of coefficient of variation:

$$\mathbb{E}[t_{\text{par}}] = \frac{T}{n_c \delta^2} (\log p)^2 + T \sqrt{\frac{(\log p)^2}{n_c \delta^2}} \sqrt{2 \log n_c} + \frac{-\log p}{n_c \delta^2}$$

**Proof of Lemma 1** Given  $(\alpha, \beta) \in \mathbb{R}_+^2$ , we write  $\forall k \in \mathbb{N}, \Delta_k = \{(T_{k+1} - t > \alpha/N) \cap (t - T_k \geq \beta/N)\}$ . We have:

$$\mathbb{P}[\Delta_{M_t}] = \sum_{k=0}^{\infty} \mathbb{P}[\Delta_{M_t} \cap \{M_t = k\}]$$

Noticing here that  $\{M_t = k\} = \{T_k \leq t < T_{k+1}\}$  we have:

$$\mathbb{P}[\Delta_{M_t}] = \sum_{k=0}^{\infty} \mathbb{P}[\Delta_k]$$

With  $\forall k \in \mathbb{N}^*, f(t_k, t_{k+1}) = N^{k+1} e^{-N t_{k+1}} t_k^{k-1} / (k-1)! \mathbb{1}_{0 < t_k < t_{k+1}}$  we get:

$$\forall k \in \mathbb{N}, P[\Delta_k] = \frac{(N(t - \beta/N))^k}{k!} e^{-Nt - \alpha} \mathbb{1}_{[0; t)}(\beta/N) \mathbb{1}_{[0; +\infty)}(\alpha)$$

and so the result announced:

$$P[(T_{M_t+1} - t > \alpha/N) \cap (t - T_{M_t} \geq \beta/N)] = e^{-\alpha} \mathbb{1}_{[0; +\infty)}(\alpha) e^{-\beta} \mathbb{1}_{[0; Nt)}(\beta)$$

**Proof of Corollary 3** From the joint probability distribution of  $T_{M_t}$  and  $T_{M_t+1}$  we have:

$$\forall (\alpha, \beta) \in \mathbb{R}_+^2, P[(T_{M_t+1} - t > \alpha/N) \cap (t - T_{M_t} \geq \beta/N)] \xrightarrow[Nt \rightarrow \infty]{\mathcal{L}} e^{-\alpha} e^{-\beta}$$

ie. :

$$N(T_{M_t+1} - t; t - T_{M_t}) \xrightarrow[Nt \rightarrow \infty]{} (Z_1, Z_2)$$

with  $Z_1$  and  $Z_2$  iid with distribution  $\mathcal{E}(1)$ . The difference between the center of the interval and  $t$  is thus the difference between two independent random variables iid with distribution  $\mathcal{E}(2)$ , which gives the desired result.

**Proof of Proposition 3** This proof comes mainly from [17].

We have  $q_{m+k} = \Lambda^{-1}(T_{m+k})$  where  $T_{m+k} \sim \Gamma_{m+k}/N$  with  $\Gamma_{m+k}$  a Gamma random distribution with parameter  $m+k$ . From the definition of  $m = \lceil -N \log p \rceil$  we get:

$$\frac{m+k}{N} \xrightarrow[N \rightarrow +\infty]{} -\log p$$

which lets us rewrite the Central Limit Theorem for the Gamma random variable as follows:

$$\sqrt{N} \frac{T_{m+k} - (-\log p)}{\sqrt{-\log p}} \xrightarrow[m \rightarrow \infty]{\mathcal{L}} \mathcal{N}(0, 1)$$

One eventually concludes by making an asymptotic expansion of  $\Lambda(q_{m+k})$  around  $q$ :

$$\Lambda(q_{m+k}) - \Lambda(q) = (q_{m+k} - q)\Lambda'(q) + o(q_{m+k} - q) = (q_{m+k} - q)f(q)/p + o(q_{m+k} - q)$$

then:

$$\sqrt{N}(q_{m+k} - q) \xrightarrow[m \rightarrow \infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f(q)^2}\right)$$

**Proof of Proposition 4** Let us write the asymptotic expansion of  $q_{m+k}$  around  $T_{m+k}$ :

$$\begin{aligned} q_{m+k} &= \Lambda^{-1}(T_{m+k}) \\ &= \Lambda^{-1}(t) + (T_{m+k} - t)(\Lambda^{-1})'(t) + \frac{(T_{m+k} - t)^2}{2}(\Lambda^{-1})''(t) + o_{\mathbb{P}}((T_{m+k} - t)^2) \\ q_{m+k} - q &= \frac{p}{f(q)}(T_{m+k} - t) + \frac{(T_{m+k} - t)^2}{2} \left(-\frac{p}{f(q)}\right) \left(1 + \frac{pf'(q)}{f(q)^2}\right) + o_{\mathbb{P}}((T_{m+k} - t)^2) \end{aligned}$$

We have:

$$E[T_{m+k} - t] = \frac{m+k}{N} - t = \frac{1}{N}(m - Nt + k) \in \left[\frac{k}{N}; \frac{k+1}{N}\right)$$

and:

$$\begin{aligned} E[(T_{m+k} - t)^2] &= \text{var}[T_{m+k}] + \left(\frac{m+k}{N} - t\right)^2 = \frac{m}{N^2} + \frac{k}{N^2} + \frac{1}{N^2}(m - Nt + k)^2 \\ E[(T_{m+k} - t)^2] &\in \frac{t}{N} + \frac{1}{N^2}[k + k^2, k + 1 + (k+1)^2] \end{aligned}$$

which concludes the proof.

**Proof of Proposition 5** Considering the approximation of a Poisson distribution by a Normal distribution:  $M_t \sim \mathcal{N}(m, m)$  and writing  $Z_{1-\alpha/2}$  the  $1 - \alpha$  % quantile of a standard Gaussian law:

$$\mathbb{P} [M_t \in [m - Z_{1-\alpha/2}\sqrt{m}, m + Z_{1-\alpha/2}\sqrt{m}]] = 1 - \alpha$$

455 We conclude by noticing that the sequence of  $(q_i)_i$  is strictly increasing.

**Proof of Proposition 6** The  $(T_{m+k})_k$  are the times of the Poisson Process with parameter  $N$  such that we know their joint probability distribution and get:

$$\forall k \in \mathbb{Z} \mid m + k > 0, \text{cov} [T_m, T_{m+k}] = \frac{m}{N^2}$$

Let us now define  $\phi : x \mapsto \Lambda^{-1}(x)$  and calculate the covariance matrix between the  $(q_{m+k})_k$ . Since  $q_{m+k} = q + (T_{m+k} - t)\phi'(t) + o(T_{m+k} - t)$  and  $\phi'(t) = p/f(q)$ , we find:

$$\text{cov} [q_m, q_{m+k}] = \left( \frac{p}{f(q)} \right)^2 \frac{m}{N^2} + o\left(\frac{1}{N}\right)$$

from which it comes:

$$N \text{cov} [q_m, q_{m+k}] \xrightarrow{N \rightarrow \infty} \frac{-p^2 \log p}{f(q)^2}$$

which concludes the proof.

**Proof of Proposition 8** We apply here the same line of argumentation as in the proof of Proposition 2, which let us approximate the Gamma distribution by a Gaussian one:

$$\frac{\Gamma_{m_0}}{N} \sim \mathcal{N}\left(\frac{m_0}{N}, \frac{m_0}{N^2}\right)$$

Let  $N_{\max}$  be the random variable of the maximum of  $n_c$  standard Gaussian variables, we have:

$$\mathbb{E} [T_{\max}] = \mathbb{E} \left[ N_{\max} \frac{\sqrt{m_0}}{N} + \frac{m_0}{N} \right] \approx \frac{m_0}{N} + \frac{\sqrt{m_0}}{N} \sqrt{2 \log n_c}$$

then

$$\mathbb{E} [N_{\text{iter}}] = \mathbb{E} [\mathbb{E} [N_{\text{iter}} \mid T_{\max}]] = N \mathbb{E} [T_{\max}] = m_0 + \sqrt{2m_0 \log n_c}$$

**Proof of Proposition 9** We have:

$$\mathbb{P} [M_{T_{\max}} \leq m] = \mathbb{P} [T_{\max} \leq T_m]$$

Furthermore,  $T_m = \Gamma_m/n_c N \sim \mathcal{N}(t, t/n_c N)$ . So

$$\mathbb{P} [T_{\max} \leq T_m] \approx \mathbb{P} [T_{\max} \leq t]$$

Finally, we seek for  $m_0$  such as:

$$\mathbb{P} [T_{\max} \leq t] \leq \alpha$$

Approximating once again Gamma laws with Gaussian distribution and using the extreme value theorem, we get:

$$\alpha = \exp \left( - \exp \left( - \sqrt{2 \log n_c} \left( \frac{Nt}{\sqrt{m_0}} - \sqrt{m_0} - b_{n_c} \right) \right) \right)$$

which concludes the proof.

**Proof of Proposition 11** Let us write  $t = -\log p$  and  $m = \lceil n_c N t \rceil$ ; we have:

$$\mathbb{P} [N_{\text{iter}} \leq k] = \mathbb{P} [M_{T_{\min}(k)} \leq m] = \mathbb{P} [T_{\min}(k) \geq T_m]$$

Furthermore,  $T_m = \Gamma_m/n_c N \sim \mathcal{N}(t, t/n_c N)$ . Denoting by  $P_{\max}$  the random variable of the maximum of  $n_c$  iid Poisson variable with parameter  $Nt$ , we have:

$$\mathbb{P} [N_{\text{iter}} \leq k] \approx \mathbb{P} [T_{\min}(k) \geq t] = \mathbb{P} [P_{\max} \leq k]$$

Finally, by doing as in the proof of proposition 2 the approximation  $\mathcal{P}(Nt) \sim \mathcal{N}(Nt, Nt)$ ,  $P_{\max}$  becomes the maximum of  $n_c$  iid Normal variables and we conclude using the extreme value theorem.



## 460 References

- [1] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja. *A first course in order statistics*, volume 54. Siam, 1992.
- [2] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis: Algorithms and Analysis*, volume 57. Springer, 2007.
- [3] S.-K. Au and J. L. Beck. A new adaptive importance sampling scheme for reliability calculations. *Structural Safety*, 21(2):135–158, 1999.
- [4] S.-K. Au and J. L. Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4):263–277, 2001.
- [5] J.-M. Bourinet, F. Deheeger, and M. Lemaire. Assessing small failure probabilities by combined subset simulation and support vector machines. *Structural Safety*, 33(6):343–353, 2011.
- 470 [6] F. Cérou, P. Del Moral, T. Furon, and A. Guyader. Sequential monte carlo for rare event estimation. *Statistics and Computing*, 22(3):795–808, 2012.
- [7] F. Cérou, P. Del Moral, F. Le Gland, P. Lezaud, et al. Genetic genealogical models in rare event analysis. *ALEA, Latin American Journal of Probability and Mathematical Statistics*, 1:181–203, 2006.
- [8] F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.
- 475 [9] P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.
- [10] A. Der Kiureghian and T. Dakessian. Multiple design points in first and second-order reliability. *Structural Safety*, 20(1):37–49, 1998.
- 480 [11] A. Der Kiureghian and M. De Stefano. Efficient algorithm for second-order reliability analysis. *Journal of engineering mechanics*, 117(12):2904–2923, 1991.
- [12] J. J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White. *Sourcebook of parallel computing*, volume 3003. Morgan Kaufmann Publishers San Francisco, 2003.
- [13] V. Dubourg, F. Deheeger, and B. Sudret. Metamodel-based importance sampling for the simulation of rare events. *arXiv preprint arXiv:1104.3476*, 2011.
- 485 [14] B. Echard, N. Gayton, and M. Lemaire. Ak-mcs: An active learning reliability method combining kriging and monte carlo simulation. *Structural Safety*, 33(2):145–154, 2011.
- [15] P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events: for insurance and finance*, volume 33. Springer, 1997.
- 490 [16] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600, 1999.
- [17] A. Guyader, N. Hengartner, and E. Matzner-Løber. Simulation and estimation of extreme quantiles and extreme probabilities. *Applied Mathematics & Optimization*, 64(2):171–196, 2011.
- [18] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- 495 [19] T. Igusa and A. Der Kiureghian. Dynamic characterization of two-degree-of-freedom equipment-structure systems. *Journal of engineering mechanics*, 111(1):1–19, 1985.
- [20] H. Kahn and T. E. Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.
- 500 [21] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing*, volume 110. Benjamin/Cummings Redwood City, 1994.

- [22] G. Last and A. Brandt. *Marked Point Processes on the real line: the dynamical approach*. Springer, 1995.
- [23] L. Li, J. Bect, and E. Vazquez. Bayesian subset simulation: a kriging-based subset simulation algorithm for the estimation of small probabilities of failure. *arXiv preprint arXiv:1207.1963*, 2012.
- 505 [24] N. Merhav and E. Sabbag. Optimal watermark embedding and detection strategies under limited detection resources. *Information Theory, IEEE Transactions on*, 54(1):255–274, 2008.
- [25] R. Rackwitz. Reliability analysis—a review and some perspectives. *Structural Safety*, 23(4):365–395, 2001.
- [26] C. Robert and G. Casella. *Monte Carlo statistical methods*, volume 319. Citeseer, 2004.
- [27] M. Rosenbluth and A. Rosenbluth. Monte carlo calculation of the average extension of molecular chains. *The*  
510 *Journal of Chemical Physics*, 23(2):356–359, 2004.
- [28] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo method*, volume 707. John Wiley & Sons, 2011.
- [29] T. J. Santner, B. J. Williams, and W. I. Notz. *The design and analysis of computer experiments*. Springer, 2003.
- 515 [30] P. Waarts. Structural reliability using finite element methods. *An appraisal of DARS: directional adaptive response surface sampling* Delft University Press, The Netherlands, 2000.